read using

Research Machines
National User Group
Journal 8

April
1984

£3·50

RML – situation grave?

EXECUTIVE COMMITTEE 1984/1985

Chairman:	Barry Mawer	206 Preston New Road, Southport, Merseyside, PR9 8NY. (0704) 24457
Secretary:	Spencer Instone	59 Avenue Road, Leamington Spa, Warwickshire, CV31 3PF (0926) 38751
Treasurer: (and Membership Secretary)	Syd Boswell	11 Westbourne Park Road, Westbourne, Bournemouth
Editor:	Delwyn McPhun	59 Avenue Road, Leamington Spa, Warwickshire, CV31 3PF (0926) 38751
Librarian:	Nick Wedd	Genetics Lab., Dept of Biochemistry South Parks Road, Oxford (0865) 511261 Ext 326
Publicity	Jim Cooley	196 Reddings Lane, Tyseley, Birmingham 11 (021) 777 7514
	Colin Watkins	B E C C, The Bordesley Centre, Camp Hill Stratford Road, Birmingham B11 1AR (021) 772 6534
	John Oakley	c/o Leicester Polytechnic, Centre for Postgrad Studies in Education, Scraptoft Campus, Leicester (0533) 461236
	Graham Townsend	Lampton School, Lampton Ave, Hounslow (01) 572 1936
	Les Watson	College of St Paul and St Mary, The Park, Cheltenham
Research Users Rep	Chris Brown	Dept of Psychology, University of Sheffield Western Bank, Sheffield S10 2TN (0742) 78555 ext 6551 or 6542

Where to go in Read Using 8

Page	Riveting article
-1	RML's 16 Bit Computer - in depth preview
0	Where to find the executive committee
1	This page intentionally left blank
2	Editors hobby horses
4	User Group Metting - a sideways look
8	Puzzle page
9	Sounding out on the 480Z
12	DONUTS - brilliant spelling game
14	Tales from the ROM - voling about in the 480Z
16	"Is that with an I or a Y?" - Sirname coding
18	Aunt Agony - all questions answered
20	LOGO programs competition
21	Circle drawing competition
23	Sorting competition - late results
24	Library management suite from Golden Hillock
27	Computers in Control - Project FLEECE
30	Random number generator in code
32	Using USR routines from BASIC
34	Disk reset function for TCL Pascal
36	Joysticks for the 380Z
38	RAM Disk of Digitimer - Review
40	Epson MX80 Low resolution graphics dump
42	Random Access Files - the state of the art
50	HELP pages
52	Research Users column
55	Software library reviews, volume 1
57	Review of library volume 2
62	Review of library volume 4
66	Internal format of BASIC programs
70	The accounts
71	Contents of past Read Usings
74	Index to past Read Usings by subject
76	Weak ending

Editors Soapbox...

Alas, gentle reader, the new RML machine is not here yet and the review contained in this issue is not entirely serious. However, it raises a lot of interesting possibilities and will, I hope, cause some questions to be asked. To my knowledge we have been told only three definite things about the next computer. It will have a 16 bit data bus (something that is supposed to make a computer inherently more powerful than 8 bit computers like the 480Z, but often doesn't). It will have a better BASIC interpreter (whatever that means). Finally, we are told that it will be OK and that we will approve. How can they possibly know?

I believe that RML's excessive secrecy is defeating its own object. Not too many years ago RML were open and frank about their plans. They also displayed complete confidence in them to the extent of giving precise dates for production. These dates were rarely met and this created a lot of ill will towards RML and (worse) very awkward questions at user group meetings. Today I sometimes wonder whether there really will be a new machine, although I cannot see even the excellent 480Z competing for very much longer in the educational market. Will the new machine have good graphics facilities as standard? Will I be able to run my existing software on it - or anyones software for that matter? Should I spend my money now on another manufacturers existing equipment, or should I wait a few months to see if RML comes up trumps? People in research may be tempted away by machines like the Apple Macintosh. For Schools there are a lot of promising new machines that would tempt my own money, particularly now that the DoI assistance is petering out.

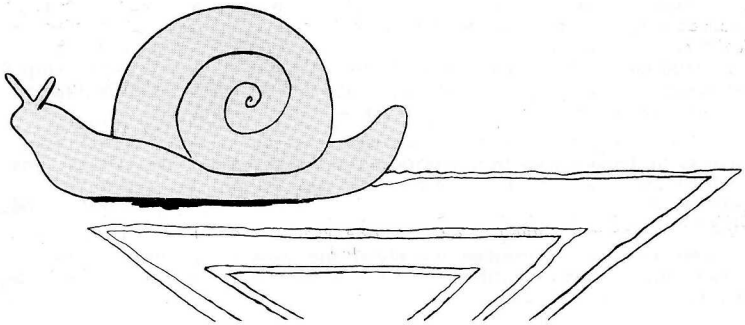
Coupled with secretiveness are the problems of deafness. It has long been a complaint of the user group that RML do not listen to users - this was brought up again at the last day meeting. Additionally, the executive committee of the group regularly meets with RML persons to represent the views of the users and to try to get things improved where there are complaints. This is a service to RML as well as their users, one would think. These meetings rarely seem to produce any effect. I presume that RML would prefer that they did not take place.

The amount of consultation with users about various aspects of future products has dwindled almost to nothing. Are there any aspects of a new BASIC that you would like to see introduced? I would want it to be fully stuctured, include data structures, module facilities for grouping common procedures and (perhaps most important) simple but powerful graphics facilities that were built into the language. It is quite likely that RML will not have thought of all of these ideas even if they would not want to implement them. RML's version of LOGO is a good example of a piece of software produced for the convenience of the producer and not the user. Like giving everyone lefthand boots because they are easier to make.

This is all heavy critism (and heavy going) which, unfortunately, I feel has to be voiced. The user group is completely ineffectual in one of its aims and maybe this is because nobody realises this. I suspect it is more likely because we haven't got a few LEA advisers on the executive committee - money speaks so much louder...

Anyway, on a different tack, I had the pleasure of seeing the new "Firmware reference manual" recently. This is a guide to the COS and ROS monitors and is chiefly for people who write assembly language programs though it will be useful addition to all programmers' shelves - not least as a good example of how to write a manual! It tells me all of those little things that I used to have to find out the hard way and manages to explain exceedingly technical stuff in a language I can understand (on the first pass!). It is full of lucid diagrams and useful tables and quick reference summaries. This manual is the long needed replacement for about 99 other manuals that all used to cross reference each other and assume that you were telepathic. Well done, who ever was responsible. I hope there's lots more where that came from!

Lastly, the plea. This is your journal - please write in it! Anything and everything will be accepted gratefully. I even have money to throw around on some articles - why not help me share it out? I would like to include a lot more that could be of use to primary schools since they are becoming a large part of the total usership. It is easiest if things arrive on a disk but don't let that stop you if it is not possible. Any letters, questions, complaints or suggestions are all welcome. The time has come to dig out that clever routine that flashed the reset light on and off and write about it!



User Group Day Meeting and A.G.M. March 28th

Your User Group Secretary is writing an account of the last day meeting not so much to tell you about the A.G.M. (which it was) nor because the Editor is leaning on him to fill up a bit of space (which he is) as to give you the flavour of a fairly typical meeting. Perhaps after reading this you may feel inclined to sally forth to our next meeting, the date and venue of which will be announced in the next Broadsheet.

10.00 - 10.30 Arrival and Coffee

Arrival in fact at Christ Church College, Canterbury which is where the meeting was held. Since much of your Executive Committee seemed to hale from the North, a number had in fact arrived in the vicinity the previous night. Judging from the looks of one or two of them, travelling to these southern parts of the United Kingdom had been the cause for some minor celebrations.

For this meeting we had arranged to meet in a very pleasant hall in the College, where Mike Sharp of the Kent Computer Advisory Centre was seen to be setting up various bits of Kent's equipment.

10.30 - 10.45 A G M

Always a trial for everyone, which perhaps explains why the Chairman did not manage to drag the meeting to order until some ten minutes after we should have started. The Chairman's address came first, in which, as ever, he attempted to justify his last term of office without sounding too self-satisfied, whilst at the same time bemoaning, as ever, the fact that RML don't seem to listen to the User Group enough. Sounds familiar? You were obviously at the last A.G.M!

Next the Treasurer's report. I always find this a challenge - can we spot the excesses of this year's postage? Has the Executive been illicitly spending the Groups funds on having a good time? What really are those 'miscellaneous expenses'?? Alack, as ever, and despite the Chairman's prompting, no questions were forthcoming.

Then the Editor's report - a sublime piece of oratory and apologia I thought - from which you may, perhaps, deduce who gave it! In fact the usual set of apologies for Read Using's late arrivals (did I say late? More like non-existent!) and then trying to blame it on the membership for not writing more articles. I expect you'll find the same pleas within the hallowed pages of this edition if you look hard enough.

There were no Constitutional amendments due to the fact that they have to be lodged with the Secretary two weeks before the A.G.M. This brilliant piece of skull-duggery basically means that you have to give amendments to the Secretary two weeks before you think of them! Good, eh what?

Nor could the Committee manage to persuade themselves that the membership subscription should be raised for the ensuing year, so it stays at a mere #10.00 per annum.

But now the high drama of the election of a new Committee. Would the Chairman's sins be visited upon his head, the Secretary's illiteracy be found out, the Treasurer's Swiss account reap the retribution it deserved? Yes, you guessed, they were all returned unopposed, together with the Librarian and a number of other committee members. Fortunately a little new blood was injected, and you will find the names of the new Committee elsewhere in this issue.

A O B could only raise an echo of the Chairman's concern over RML's deafness to the siren calls of the User Group, so we moved on (late as ever) to the RML presentation.

10.45 - 11.45 **R.M.L. to reveal all**

Well, 11.05 would describe the starting time better, but we had the pleasure of a new face from RML in the person of Mike Harper, a new marketing manager. A new and pleasant style of presentation, too. Gone the days of heavy persuasion, in with a new wind of realism. Not since the early RML days when Mike O'Regan tried hard to persuade myself and other teachers that the RML computers were 'not really at all suitable for schools' had we heard such transparent truths! Your secreatry was so dumbfounded that his usual skill at note-taking left him, so I can only give you the following very poor summary of what Mike said.

Releases

380Z stand alone Winchesters (not, as you may think, large glass bottles, but hard disk systems of incredible capacity and power!)

A chopped down 380Z fitted inside what looks like a 480Z disk drive, no disks, but very suitable for control, and costing less than #1000.00

The long sought after (and incredibly costly) 64K ROM pack

Pluto boards for the 380Z (called such because apparently anyone can now emulate Walt Disney)

Silicon disk for the 480Z for a (wait for it) special introductory price of #125.00

An amazing design package at a cost of #500.00 per network

I'm sure there were lots of other things which I missed, partly because the audience got very restless over philological distinctions applied to the word 'cheap' (i.e. schools think that #500 is not cheap and RML think that #500 is cheap compared with what this amazing design package cost before!)

I'm afraid that another reason why I missed more of the 'RML in the blue corner, Users in the red' bit, is that discussion was so interesting (not to say prolonged) that by now lunch looked like disappearing into tea-time, so the kitchen staff had to be visited and placated. However, if you want to come and let your views known to RML there's certainly no more open forum than at one of the User Group's Day meetings!

11.45 - 12.45 Richard Ennals - 'Micro-Prolog'

Only half an hour late by now, we went into a masterly talk about Micro-Prolog by one of its foremost protagonists, and ex-History teacher, Richard Ennals. Richard talked about using Prolog in the classroom, and what children could actually achieve in the classroom with the language. Not a lecture from cloud-cuckoo land either, since Richard came hot-foot from the various classrooms in which he is conducting his research project.

Amongst other things, he showed that the language is quite capable of proving that your Chairman deserves damnation, as being unrepentant over various mortal sins, and that all of you (including your Chiarman) love Mary. Lucky girl is all I can say. I shall spoil the talk no more, except by saying that we hope that Richard will be able to write up his talk as an article for the next issue of Read Using, so that he may enlighten a wider audience about the merits of Prolog.

12.45 - 1.45 Lunch

Carefully provided by the College at meagre cost only 50 or so steps from the lecture hall. Those who had not booked with the Secretary had to forage around Canterbury in the rain. To the three of you who booked a meal and then never turned up, your Secretary sends his rebukes - what an example to set your pupils! As it happens, this enabled the Secretary to feed 3 other starving folk who had forgotten to book!

1.45 - 2.45 Richard Noss - 'Using LOGO in the classroom'

We had all looked forward to a ding-dong battle between the two linguistic protagonists at the meeting. Richard of the Ennals had told us that Prolog was declarative, and LOGO procedural. That seemed to some of us certainly like fighting talk, and we looked forward to hearing Richard of the Noss defend himself from such an ugly smear. Alack aday, it didn't turn out like that at all, and both the Dicks (if they will permit me such intimacy) were most polite to each other. This, I may say, was rather different to the language they both used about BASIC, which they were not polite about at all. Some of us who have used a good structured BASIC (I wonder which) felt this a little unfair, but never mind.

As it was Richard attempted to talk to us about LOGO without once touching the computer. I am glad to say that this Luddite approach to the language soon failed thanks to comsumer demand! Richard had some very nice things to say about LOGO, some nice things to say about the RML implementation, and a few nasty things to say about the RML version. Again, I am not going to forestall a future article and review of RML LOGO which we hope to get into a future issue of Read Using.

2.45 - 3.45 David Doyle - 'Motors do it digitally'

This disgraceful title was indeed a prelude to a talk about 'turning on', albeit started, through no fault of David, some 30 minutes late. I always feel particularly vulnerable and ignorant in the face of these electronics wizards, I don't know about you. There is something about the ease with which they handle a Dartington Pear, or whatever it is, that I find distinctly disquieting. And I'm never quite sure of the distinction between A2D, D2A, and R2D2. Never mind!

David did manage to convey a little to this ignoramus of what can be simply done with your 380Z, a few motors, and a few kids (all of whom will no doubt grow up and increase my inferiority complex). David's talk, one of many I inwardly panicked my way through, was in fact most reassuring. Without mentioning R2D2 once (or any of those other mysteries) he managed to convey that you can do useful, interesting and even exciting projects with only a little equipment.

David comes from the ILEA Design and Technology Centre, so by the time you read this, when 'Red Ken' will probably have been thrown out on his neck by 'the Great she-elephant of Finchley', I expect he'll be looking for a job. I suggest any schools out there wanting their traffic lights repaired, or 'Big-Trak's re-motored get in touch with him direct!

4.15 Hometime

Only half an hour late, we break up, after a long but worthwhile day. Not every day meeting is as good as this one (and certainly not all are as well attended). But next time one of those funny notices comes through to your pigeon-hole, why not give it a whirl and come along. You have nothing to lose but your sanity!

Spencer the Secretary.

WILKINS DISAPPROVED OF USING DISKS FOR MARTIAL ARTS.

Puzzle Page

Hidden in the square are thirty 'everyday' computer terms. The words could be written forwards, backwards, diagonally, etc.
The words are :-

ROM	PIP	CPM	CPU
DISK	ASCII	INPUT	ERROR
COBAL	RESET	BASIC	LOGIC
OUTPUT	BINARY	MEMORY	NUMBER
PASCAL	PROGRAM	MONITOR	BOOLEAN
NETWORK	SOFTWARE	HARDWARE	ASSEMBLY
KEYBOARD	COMPUTER	PROCEDURE	LANGUAGES
ALGORITHM	SUBROUTINE		

How many can you find ?

P C W X C P U V I I C S A N F N C S B D
M T L G O X Y M C P R S L A B O C R T F
H M A R G O R P I E O L D O I N P U T G
L O G I C M R V A E B E R A W D R A H D
Y N W H T E Z L L K I C I S A B P Z F N
S U H G S F Q C G S E G A U G N A L R U
J P Q E G X G S O L R Y R A N I B G T O
P A T C I S U B R O U T I N E V D F C T
B A T N Y S E N I H H G R P X W C T D A
Q K S V U X R O T M E M O R Y O Y V R I
J V Y C I X N D H U P S M I M G U X A V
I F S D A A V R M O M P U P D O Z I O Y
L J G W E L B G T R C H U P S I B N B L
L B D L O N N S O X M T S A E T S E Y B
D O O P M O R U M U E E G R Y R J K E M
F O X I X S I R M R E W M O R E R I K E
B W S P M N R V N B M E R A W T F O S S
W I Z J O H O B J N E Q T U P T U O R S
M O Q M U F L P R O C R D U R E B Y H A
C P P L B N E T W O R K E U R S F E H F

Feel free to photocopy this page and bore your computer class with it!

Sounding out on the 480Z

Many sensible machines provide the means of playing notes which have some useful musical relationship to each other. After graphics, sound is the next most interesting what-can-I-make-it-do-Miss? for kids and would provide a valuable incentive to learning about computers and programming. Would, that is, if you didn't have to type PUT statements with sequence introducers and CP's and random numbers. And all that to get an out of tune 'meep!'.

Here are a couple of routines that help the situation a bit. The first simply enables you to predict the note that is meeped. The subroutine below is all you need.

```
1000 REM --- Play note number N with duration L
1010 PUT 27,64,242/2^(N/11.9),L,65,7
1020 RETURN
```

This plays a note for a certain length of time. The pitch of the note is given by the note number, N. You control the length of time that the note is played for with the variable L. You could use the subroutine by simply setting N and L to the note and length that you want and then calling the routine. For instance:

```
N=0 : L=50 : GOSUB 1000
```

would do the trick.

The reason why line 1010 is so horrible is that the note number, N, corresponds to the notes on a piano keyboard (or any similar scale). The lowest note that you can have is C natural an octave below middle C. This is the note played when N=0. The notes go up in semitones, that is, N=1 plays one semitone above N=0 or C sharp in fact. N=2 gives D natural, a whole tone above N=0. Since there are twelve semitones to a octave, playing note number 12 will play middle C. Now "copy and complete the following table"...

C	0	C	12	C	24	C	36
C#	1	C#	13	C#			
D	2	D		D			
Eb	3	Eb					
E	4						
:		:					
Bb	10						
B	11	B	23	B	35	B	47

The notes start to lose their accuracy if you use very large note numbers but there's a lot you can do with four octaves. Here is a simple program that plays notes that it reads with their durations from a list contained in data statements. You must have already typed in the subroutine above for this to work.

```
10 READ N, L            :REM  Read note number and length
15 IF L=0 THEN END      :REM  Stop when the length is 0
20 GOSUB 1000           :REM  Play the note
25 GOTO 10
30 :
50 DATA 5,40, 5,40, 12,40, 12,40, 14,20, 16,20, 17,20, 14,20, 12,80
60 DATA 10,40, 10,40, 9,40, 9,40, 7,40, 7,40, 5,80, 12,40, 12,20, 12,20
70 DATA 10,40, 10,40, 9,40, 9,20, 9,20, 7,60, 7,20, 12,40, 12,20, 12,20
80 DATA 10,20, 12,20, 14,20, 10,20, 9,40, 7,20, 7,20, 5,80, 0,0
```

The second routine is rather heavier going. It enables noises to be produced varying from high pitched hissing to low pitched raspberries. With some playing around, chuff chuffs, explosions, revving motorbikes, landing airplanes and the like could adorn your programs.

The subroutine and its associated data will be more of a chore to type in and it all must be right! The REM statements after the data are not important, unless you can understand them then you probably wouldn't want to anyway. The subroutine at line 1000 pokes in some machine code to generate a noise. Boffins will realise that this subroutine is more complex than it need be because it reads the machine code data in the form of hexadecimal characters rather than numbers but I shouldn't worry about that. Lines 10 to 40 **must** be run before the noise routine can be used and so they should appear near the beginning of your program.

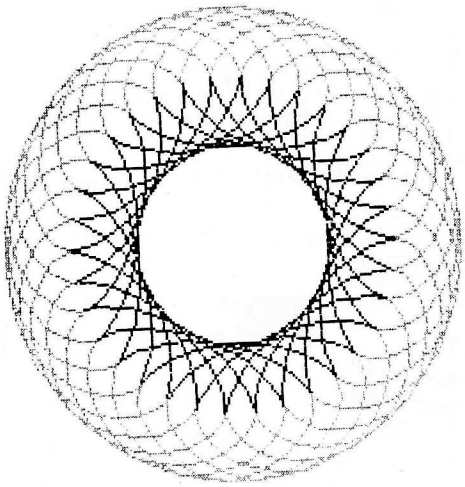
Thereafter, typing CALL NZ, P, D will make a noise. P controls the Pitch of the noise and D controls the Duration of it. The duration of the actual noise in fact depends on the pitch as well as the D variable. The NZ variable is set up to be the address of the machine code noise routine. It is vital that you do not change the value of NZ after it has been set up since this will instantly result in whole worlds falling about your ears.

```
10 REM --- Nasty noises for 480Z
20 CLEAR 100,,50
30 NZ=PEEK(&11C)+PEEK(&11D)*256+1        : REM  NZ := beginning of cache area
40 AD=NZ : GOSUB 1000                    : REM  AD is the place to poke code
50 :
60 REM      CALL  NZ,  Pitch,   Duration
70 REM
80 REM  Pitch     0 -> 255
90 REM  Duration 100 to 4000
100 :
110 CALL NZ, 5, 4000     : REM  Some example noises!
120 CALL NZ, 50, 2000
130 CALL NZ, 120, 1000
140 CALL NZ, 255, 500
150 FOR I=1 TO 140 STEP 5
160  CALL NZ, I, 500-I
170 NEXT I
180 END
200 :
1000 REM --- Poke code from data to memory at AD
1005 DEF FNHX(A)=-(A<65)*(A-48)-(A>64)*(A-55)
1010 READ A$: IF A$="Z" THEN RETURN
1020 FOR I=1 TO LEN(A$) STEP 2
1030  IF MID$(A$,I,1)=" " THEN 1080
1040  H=FNHX(ASC(MID$(A$,I,1)))
1050  L=FNHX(ASC(MID$(A$,I+1,1)))
1060  POKE AD,H*16+L
1070  AD=AD+1
1080 NEXT I: GOTO 1010
1090 :
1100 DATA 79        :REM  LD  A, C            ; Check for 2 stacked
1110 DATA FE02      :REM  CP  2               ; numbers
1120 DATA C0        :REM  RET NZ
1130 :
1140 DATA 2100E8    :REM  LD  HL, E800H       ; Start of random data
1150 DATA D1        :REM  POP DE              ; Duration
```

480Z Soundz

```
1160 :
1170 DATA 3A02FF    :REM  LD  A, (CMASK2)     ; Get control port 2 mask
1180 DATA E6DF      :REM  AND 11011111B
1190 DATA 4F        :REM  LD  C, A
1200 DATA 7E        :REM  LD  A,(HL)          ; Get random bit
1210 DATA E620      :REM  AND 00100000B
1220 DATA B1        :REM  OR  C
1230 DATA 3202FF    :REM  LD  (CMASK2), A     ; Update mask, port
1240 DATA 0E1A      :REM  LD  C, CPORT2
1250 DATA ED79      :REM  OUT (C), A
1260 DATA 23        :REM  INC HL              ; Next byte
1270 :
1280 DATA C1        :REM  POP BC              ; Wait for stacked delay
1290 DATA C5        :REM  PUSH BC
1300 DATA 41        :REM  LD  B, C
1310 DATA 10FE      :REM  DJNZ $
1320 :
1330 DATA 1B        :REM  DEC DE              ; Repeat till DE=0
1340 DATA 7A        :REM  LD  A, D
1350 DATA B3        :REM  OR  E
1360 DATA 20E4      :REM  JR  NZ, -27
1370 :
1380 DATA C1        :REM  POP BC              ; Discard delay and return
1390 DATA C9        :REM  RET
1400 DATA Z
```

Ed.



DONUTS

Donuts was written by two fifth form pupils at Golden Hillock School as an exercise for pupils with spelling difficulties. It runs in BASIC Version 5 with COS 4. Donuts is available from the user group library.

The game combines the challenge of arcade type games with the need to know the alphabet and be able to spell. It has been used with enthusiasm by the remedial department but primary purists have pointed out that lower case letters should be used instead of capitals. To change to lower case the data words in lines 840 onwards need to be retyped in lower case, and line 370 should be changed to: PLOT A+6, 52, 97+CH

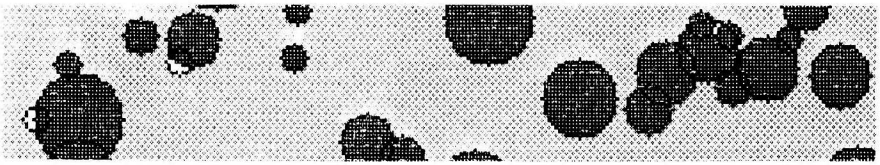
The game is played on 5 levels, the words inrease in size and the game gets faster. The delays for each level are stored as data in line 890.

Jim Cooley

```
10 REM WRITTEN BY IAN EDWARDS
20 REM                AND DEAN KELLY
30 REM of Golden Hillock School -
40 REM      Birmingham.
50 PUT12
60 DIM NT$(100), W(20)
70 RANDOMIZE
80 LET J$=" "
90 GOSUB 680
100 FOR LP=1 TO 50
110 READ NT$:NT$(LP)=NT$
120 NEXT LP
130 FOR LP=1 TO 5
140 READ DL(LP)
150 NEXT LP
160 INPUT"What level 1-5";LV
170 IF LV<1 OR LV>5 THEN INPUT LV:GOTO 170
180 ON ERROR GOTO 660:ON BREAK GOTO 660
190 ?"    PRESS RETURN":INPUT QE$
200 LET LV=LV-1
210 LET JI=INT(RND(1)*9)
220 LET N$=NT$((LV*10)+JI)
230 PUT12
240 PLOT 20,20,N$
250 INPUT" Press return to start ";GH$:PUT12
260 LET LL=LEN(N$)
270 FOR LP=1 TO LL
280 LET W(LP)=ASC(MID$(N$,LP,1))
290 NEXT LP
300 NULL2,11
310 LET LP=1
320 LET S=0
330 LET X=40
340 LET A=INT(RND(1)*75)
350 LET CH=INT((A/75)*26)
360 PLOT X-I,0," "
```

cont...

```
370 PLOT A+6,52,65+CH
380 PLOT 14,58,"                              "
390 FOR TW=1 TO DL(LV+1):NEXT TW
400 PRINT
410 PLOT 14,58,LEFT$(N$,LP-1)
420 LET S=S+.5
430 IF POINTS(X,0)=W(LP) THEN LET LP=LP+1:PUT12:GOTO 340
440 LET RF=POINTS(X,0)
450 IF POINTS(X,0)<>128 AND POINTS(X,0)<>32 THEN GOTO 550
460 PLOT X,0,7
470 LET I$=GET$(0)
480 IF I$<>"" THEN J$=I$
490 IF J$="0" THEN LET I=1
500 IF J$="1" THEN LET I=-1
510 IF J$=" " THEN LET I=0
520 IF X<1 OR X>78 THEN LET J$=" "
530 LET X=X+I
540 GOTO 340
550 NULL 0,0
560 PUT 31,13,13,
570 PRINT"YOU ARE DEAD"
580 IF LP<=LL THEN PRINT"You failed to score ":GOTO 610
590 PRINT"YOU SCORED:";INT (S)
600 SC(LV)=INT(S)
610 PRINT"WOULD YOU LIKE TO PLAY A GAME ?"
620 LET Z1$=GET$(0)
630 IF Z1$="Y"OR Z1$="y"  THEN GOTO 160
640 IF Z1$="N" OR Z1$="n" THEN 660
650 GOTO 620
660 NULL 0,0
670 END
680 NULL0,0:PUT12,31
690 PRINT
700 PRINT
710 PRINT
720 PRINT"  THE aim of this game is to construct  the word which will appear in
the centreof the screen.Random letters will be"
730 PRINT "coming down the screen.You must eat all the letters in order to survi
ve."
740 PRINT"  When you have completed the word,you  must dodge all the letters to
increase  your score.If you do not complete the"
750 PRINT"word,you will not score.You have only   one life-use it well."
760 PRINT
770 PRINT"  YOU WILL DIE IF YOU HIT THE SIDES !"
780 PRINT
790 ?:?:?" 1 To move left - 0 To move right"
800 ?
810 ?"            SPACE to stop"
820 ?
830 RETURN
840 DATA WAS,AND,ALL,BUT,ARE,FOR,HAD,HIM,HIS,NOT
850 DATA SAID,THAT,HAVE,THEY,WITH,BACK,CAME,DOWN,FROM,WERE
860 DATA ABOUT,FIRST,LITTLE,BEFORE,RIGHT,WHICH,WHERE,SHORT,OTHER,BELOW
870 DATA SIMPLY,PURCHASE,TELEPHONE,EXAMPLE,RHUBARB,FURTHER,CONTACT,VALUABLE,ACCU
RATE,SCHOOL
880 DATA REASONABLE,EFFORT,CAULIFLOWER,GLOBAL,MAGNIFICENT,OPPORTUNITY,TELEPHONE,
TELEKINESIS,CATASTROPHIC,SARCASTIC
890 DATA 70,50,25,10,1
```



Testing for pressed keys - auto repeat - and all that.

Thanks to the way that the 480Z ROS handles the keyboard you can check to see what keys are being pressed at any time. This is, of course, undocumented. If you have used seven undocumented features before breakfast...
Being able to test for pressed keys makes it possible to simulate 'auto-repeat' in programs. Since all of the keys can be checked then you can make use of the SHIFT, CTRL and CAPS keys in programs. Also, there is no longer any reason why SHIFT and CTRL should be the only 'key modifiers' in a program. To understand how this is possible you need to know how ROS reacts to the savage key pressing user. If you don't care how it works but just want to use it then skip this bit.

When a key is pressed it generates an 'irritation'. This causes the micro-processor to stop whatever it is doing, quickly note what key has been pressed and then go back to whatever it was doing as if nothing had happened. This is called 'processing the irritation'. A suitable analogy might be the processing of mosquitoes while trying to read a book. Once the mosquito lands it must be quickly processed. You stop reading, swat madly about for a bit, and when the immediate danger has gone, you return to your reading where you left off. I believe that the term 'irritation' originated from the well known habit machine code programmers have of crashing their computers. This happens because m/c programmers tend to type in very fast short bursts. The poor micro ends up trying to process hundreds of keys at once and not only gets irritated but goes into a 'rather annoyed' state and refuses to talk to the programmer any more. Only the short sharp prod of the reset button will bring the micro back into line again.

Anyway, an irritation is generated when you press down a key and ROS notes that the key is currently pressed down. When you release the key another irritation occurs and ROS notes that the key is no longer pressed down. If you make a long beep (say by typing PUT 27,64,100,255,65,7 in BASIC) then you can hear irritations being swatted if you press and release keys while beeps are going.

The information about which keys are presently pressed is stored in an 8 byte 'bit-map'. This is 8 bytes in which each bit represents a key on the keyboard. But there are 65 keys! comes the anguished cry - we'll come to that. The position of the bit in the map that represents a particular key is determined by a very good random number generator hidden deep in the heart of ROS. This is so that if you discover the bit map in the first place, it will then take you ages trying to find out how to use it (actually, there may be a better reason). Perhaps ROS in fact stands for Really Obscure Software! The bit map, by the way, starts at FF7F hex.

The **simple program** that follows tests each key in turn to see if it is being pressed. If it is pressed then the number of that key is printed. It takes a couple of seconds to go through each key so be patient.

The subroutine starting at line 100 tests to see if key number N is pressed. The table that follows the program gives the numbers that correspond to various keys. The routine returns -1 if the key is pressed and 0 if not. This subroutine could be incorporated into a program. For example, you might have a program that uses the keys U, D, L and R to move something Up, Down, Left and Right. Why not make these keys auto-repeat, that is, so long as the key is pressed the action, whatever that might be, happens. Instead of using a GET statement to get a character, and then testing for U, D, etc, call the magic subroutine with N equal to the number of the key that you want to

test. If this returns A with a non-zero value then you know the key is being pressed.

```
10 FOR N=0 TO 63
20   GOSUB 100
30   IF A THEN ?N
40 NEXT N
50 GOTO 10
60 :
100 REM --- Test key number N, return A as flag
110   X=INT(N/8):Y=(N/8-X)*8
120   A=(PEEK(&FF7F+X) AND INT(2↑Y+.5))>0
130 RETURN
```

Key	Number		Key	Number
Arrow keys:			Alphabet:	
Left	46		A	45
Right	22		B	10
Up	30		C	9
Down	14		D	41
			E	57
Function keys:			F	34
F1	62		G	42
F2	54		H	36
F3	6		I	16
F4	38		J	44
			K	32
Control keys:			L	40
RETURN	59		M	0
SPACE	4		N	12
REPEAT	12		O	56
ESC	61		P	23
CTRL	5		Q	21
SHIFT	3	(both keys)	R	18
CAPS LOCK	37		S	33
DELETE	51		T	58
LINE FEED	43		U	60
			V	2
Numbers:			W	17
0	48		X	1
1	29		Y	20
2	53		Z	13
3	25			
4	49		Other: /	15
5	26		.	7
6	50		,	8
7	28		@	63
8	52		;	39
9	24		:	47

Two of the features of the 480Z that **are** documented have been removed from 480Zs that are being produced these days! I discovered this when a program that generated sounds using the Digital to Analogue Converter did not work on a new 480Z. After a considerable amount of senseless hair and wire pulling (clumps of both came out in my hands) I noticed that the relevant chips were not there! The same goes for the spare CTC chip. No doubt this saves RML some money which could perhaps be spent on un-documenting these features. A mention in their newsletter would have saved a lot of hair.

[The Vole]
ex of Cheltenham

Is that with an I or a Y?

It is a problem that I'm sure our secretary has and I suspect many more. You are given a name over the telephone, or it's scrawled in an indecipherable hand-writing on paper. It is Smith, Smyth, Smythe or even Smitty ? How are you going to look the name up in a file, if you can't even read past the second letter? There is a solution and it comes from an age when bugs crawled around on the floor and boots were something you put on the end of your legs. It was originally developed for the 1890 American census. Called 'Soundex', it transforms a given name into a code, consisting of one letter and three digits. Hopefully all variations of a name, will have the same code. Therefore matching this code with the codes of all the names in the file, will give you a short list to choose from, or even a unique solution.

There are four steps to generate the soundex code

1) Keeping the first letter of the name, remove all occurrences of a, e, i, o, u, w and y.

2) For each remaining letter after the first, assign a particular number (see program for actual number).

3) If two or more letters with the same code are adjacent in the original name (before step 1) then omit all but the first.

4) If the result is less than four characters long, pad it out with zeros.

For example:

Name	Code	Name	Code
Smith	S530	Rogers	R252
Smyth	S530	Rodgers	R325
Smythe	S530	Sinclair	S524
Smitty	S530	St.Clair	S324

As you can see, although the system works well with the Smiths and most names, there are variations of some common names that give different codes. But this is only the most simple of techniques. It can be enhanced by adding the initial of the first name, or the date of birth, the sex, or any other individual characteristic.

And here is the program in BASIC

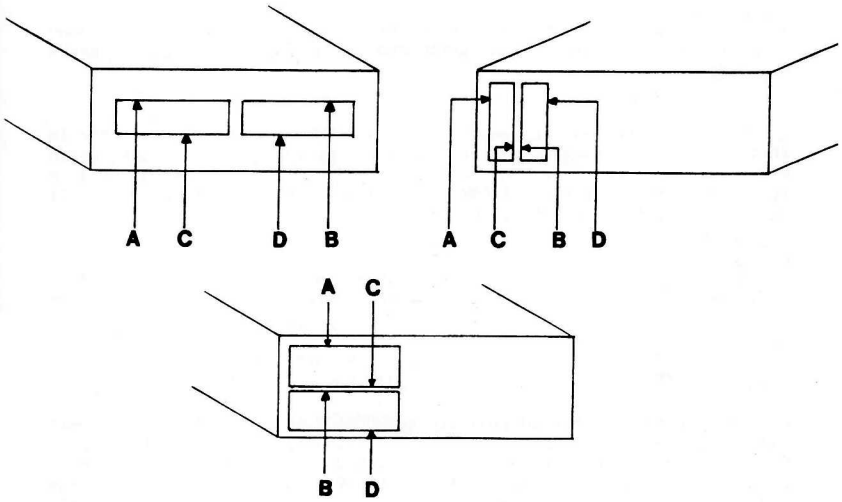
```
100 REM  Soundex subroutine + driver
110 Input "Input last name (Upper case)";N$
120 GOSUB 200
130 PRINT "Soundex code = ";S$
140 GOTO 110
150 END
```

continued....

```
200 REM Start of soundex subroutine
240 S$=MID$(N$,1,1)                          'Get first letter
245 L$=S$
250 IF LEN(N$)=1 THEN 400                    'Check for short names
260 FOR I= 2 TO LEN(N$)                      'For each letter
270  E$=MID$(N$,I,1)                        'Get letter
280  E=ASC(E$)-64                            'Convert to 1-26
290  IF E>26 OR E<1 THEN 370                 'Miss non letters
300  K$=MID$("01230120022455012623010202",E,1)  'Get soundex code
310  REM      ABCDEFGHIJKLMNOPQRSTUVWXYZ
320  IF K$=L$ OR K$="0" THEN 360             'Miss sound-alikes
340  S$=S$+K$                                'Build up code
350  IF LEN(S$)>3 THEN 400                   'Only the first four are needed
360  L$=K$
370 NEXT I
400 S$=LEFT$(S$+"000",4)                     'Pad if needed
410 RETURN
420 END
```

Yours individually,

T216



Illogical Disk Drives...

Dear Aunt Agony,

Thanks to growing demand the 'Agony Aunt' column continues. A team of crack boffins will answer all those questions that Tech Support won't answer or are far too embarrassing to ask in the first place! Questions written down please and addressed to the Editor of Read Using.

Q: **When do I use " " round a filename and when don't I?**

A: In High Level languages, such as BASIC, a collection of characters (known as a string) is usually delimited in some way. In BASIC, the delimiters are the double quote markers ". Commands such as LOAD, take a string as an argument, and therefore require quotation marks round the filename. This also applies to using data files inside a program. Other high level languages such as Fortran, Pascal and Algol have similar conventions, although some use the single-quote character. Programs which use filenames, but are not languages (such as the text editors TXED and WordStar), don't require the " character, as they are always dealing with strings.

Just to add more confusion: at any time when you are using a filename from CP/M such as when you say **BASIC PROG**, you never use quotation marks round PROG.

Q: **We have started to use QUEST on our 380Z. One problem we have is that in order to change typeface, skip perforations etc. for our Epson MX80 we have to dive into BASIC. What we need is a command program (probably accessed through TURNKEY) which on completion of options will automatically fire up QUEST. Can you help?**

A: The best way to do this is to use the SUBMIT program, which is to be found on the CP/M distribution disc. Create a file, with the extension **.SUB** which contains the names of the programs you wish to run. For example:

```
BASIC OPTIONS          [OPTIONS is the BASIC Program]
QUEST
```

Now, if you type **SUBMIT QOPTION** (where QOPTION the name of your submit file), you will find that BASIC is automatically loaded and the program OPTIONS is run, and that when it is finished QUEST will then be loaded. Note that keyboard input (using INPUT or GET) in the OPTIONS program will still work. You can now use TURNKEY to automatically do SUBMIT QOPTION when you type B.

Q: **In BASIC 6 can I print using a string variable after the USING?**

A: Yes, you can follow USING either with a string expression or a line number. A string expression can be letters in quotation marks, string variables, string functions, or combinations of those. For example:

```
100 A$="Amount ####.##"
110 PRINT USING A$, V1
120 PRINT USING A$, 43.3
```

Q: **Is it possible to clear the High Resolution Graphics off the screen after leaving BASIC? I often load up TXED to edit a file and the realise that I can't actually see it because of the picture on the screen! Having to press the reset button at this point is a bind.**

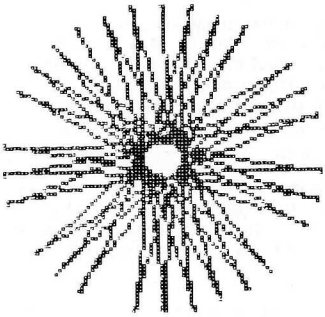
A: Yes. All you have to do is enter the front panel and poke around a bit. The procedure is different on the 380Z and 480Z. First on the 380Z, move to location FB00 and put an 8 there. On the 480Z put a 4 to I/O port 3A. What you actually type once in front panel is this:

For 380Z:
```
M FB00 (return)
8 (return)
K
```

For 480Z:
```
3A : 4 , K          (the colon and comma are not misprints!)
```

Q: **Somebody told me that if you have a picture on the HRG screen and the RESET button is pressed, you don't lose the picture. I have tried this but my screen goes blank. How do I stop this?**

A: This seems to be the issue for HRG problems. When you press the reset button, all the text on the screen is cleared. It also seems as if the HRG screen is cleared. In fact only the colour lookup table is cleared. What this means is that all the colours are set to black, so you can't see the picture. If you go back into BASIC by a J>103 command, and type **CALL"COLOUR"** the picture will magically reappear.



COMP

LOGO Programs

Most people who have used LOGO seem to get that glazed-eye look that you used to see in the old days of computing. They also seem to acquire a missionary zeal that exceeds that of the Moonies! But Read Using has had only the tiniest affair with LOGO, to whit 2 programs written by a semi-anonymous 'Cathy' published in edition 7.

We shall now put an end to all this ignorance, and in the best tradition of those who are not quite sure what they are trying to do, we are pleased to announce a Grand Competition.

Read Using's readers, and for those who are teachers, their pupils, are invited to submit any LOGO program to this competition. The program must run under RML's version of LOGO, and will be judged on the merits of the output it produces. We do not want to tie down competitors by being so strict as to specify the criteria that will be used to judge 'merit', mainly because we don't know what we'll use. But you can be sure that the Editor's decision is final, and that's that! One thing is, however, certain: speed will not be a factor taken into consideration!

Any program that is submitted may be put into the User Group Library, unless the person who submits it requests otherwise.

To try to encourage our younger readers (are there any?) prizes will be awarded in three categories based on age ...

1. Programmers who on 1st Sept 1984 will be 11 years old or less

2. Programmers who on 1st Sept 1984 will be between 12 and 16

3. The rest of you!

There will be an additional prize awardable to persons of any age for the best LOGO program whose output is in the main non-graphical (like poetry and fings!). Of course, this prize could be awarded in addition to one of the winners in the above three categories (see, even Read Using has heard that LOGO isn't just a load of old Turtles). The prize in each category will be books, software or even ready money up to a value of £10.00

In case of any dispute, or needed clarification of the rules, or what is meant by 'best' the Editor's ruling stands and there's no appeal! Entries must be received by July 31st 1984 at the latest, and the results will be published in RU 10. So get to those keyboards, and let's see if all this nonsense about 'the child as epistomologist*' means anything.

* For the ignorant, see 'Mindstorms' by Seymour Papert, page 19.

ETITION

Circular arguments

For those of you who find drawing pretty pictures using LOGO far too trivial, here's what we think you will find a more challenging competition.

Entries are invited for the fastest circle-drawing program in HRG. The program must be capable of drawing a circle of any radius, at any centre. To be kind we will only test your programs with circles that fall entirely within the HRG screen, and to be absurdly kind to competitors in the 'other languages' category, we shall restrict the radius and the co-ordinates of the centre to whole numbers (becoming almost trivial, eh what?).

Because RML's have this nasty habit of outputting points and straight lines to the screen, instead of nice smooth lines, there arises the problem of when is a circle not a circle. Yes, you guessed it - the Editor will decide. Remembering that the largest circle your program will be asked to produce will still fit on the screen, a circle of this size must still look circular - a rather overgrown octogon will just not do!

In anticipation of an absurdly high standard of entry, with correspondingly 'don't blink now, you've missed it' speeds, all programs as entered must be able to be timed - in other words, you must draw your circle 10 times or 100 times etc so that an editorial stopwatch can be brought to bear. And in the case of 'other languages' the Editor must be able to satisfy himself that the circle **has** been drawn the claimed number of times.

Having talked about these two categories, let's make it clear that entries will be judged in two classes, and that you may, if you wish, specify whether your program should be used on a 480Z or a 380Z. First prize in each category will be software to the value of £20.00 from Software Production Associates.

1. Entries written in any dialect of BASIC which has been released by Research Machines

2. Entries written in any other language

Your programs will be tested drawing circles of various sizes, and, if necessary, the average time taken to draw a circle will be used, taken from timing circles of different sizes. As with the LOGO competition, all your entries may be placed in the User Group Library unless you specifically request otherwise.

The result of the competition will be announced in Read Using 10, and entries must be received by the Editor on disk or cassette by July 31st 1984 at the latest.

continued...

Following a long line of Read Using Competition precedents, here is a BASIC program which is so slow anyone ought to be able to beat it.

```
 10 INPUT "Radius and co-ordinates of centre please"; R, X, Y
 20 INPUT "How many times shall I draw the circle for you"; N
 25 CALL"RESOLUTION",0,2
 30 PRINT "Started..."
 40 FOR I = 1 TO N
 50   PI = 3.14159
 60   CALL "PLOT", X + R, Y, 1
 70   FOR Z = 1 TO 360
 80     A = Z/180*PI
 90     CALL "LINE", X + R*COS(A), Y + R*SIN(A)
100   NEXT Z
110 NEXT I
120 PRINT "Finished"
```

Good luck; see you around! - Editor

Last weeks puzzle unpuzzled - here is that strange building again, suitably monogrammed.

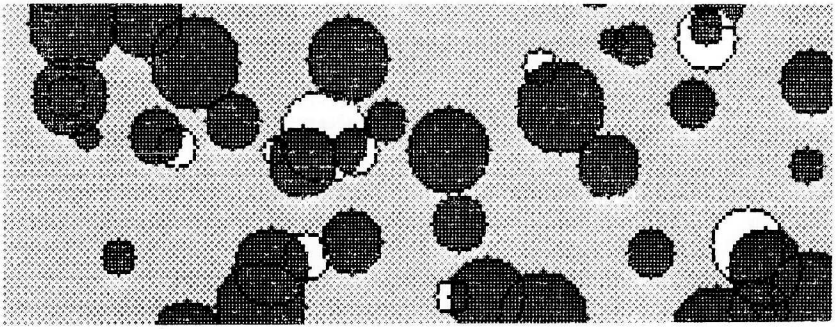
Late Sorting Results

Yet another entry for the pre-historic sorting competition has come to light. This interesting sort was submitted by David King of ILECC. It uses an Address Calculation (Hashing, Pigeonholing) method whatever that might be. Anyway, for short files it is remarkably fast for so simple a program. So fast, in fact, that it is faster than the Nick Wedds prizewinning entry announced in the last issue! A consolation prize will be hurriedly dished out. Here are the timings for various length files:

Number of items	Mean Sorting time
128	8.8
256	20.1
512	120.0

If anyone would like to write an article explaing Pigeon Hash Calculations...

```
10 CLEAR 32000 : DIM D$(512),P(720)
20 ? : ?"ACPOINT -- address calculation sort"
25 ?"with pointers"
30 ? : INPUT "Input filename"; IP$
35 OP$=IP$+"AP" : IP$="C:"+IP$
40 IF NOT LOOKUP(IP$) THEN ? : ?"Can't find ";IP$;"?" : GOTO 30
60 IF LOOKUP(OP$) THEN ERASE OP$
70 RESET : CREATE OP$ : OPEN IP$ : ON EOF GOTO 90 : C=1
80 INPUT LINE #10, D$(C) :  C=C+1 : GOTO 80
90 N=C-1 : ? : ?N;" items read from ";IP$
100 ? : ?"Press any key to start sort:"; : J=GET() : ?
105 FORI=0TO720:P(I)=-1:NEXTI
110 FORI=1 TO N
130 X=ASC(D$(I))*27:IFLEN(D$(I))>1THENX=X+ASC(MID$(D$(I),2))-64
135 X=(X-1755):Z=I
140 IFP(X)=-1THENP(X)=Z:GOTO145
            ELSEIFD$(P(X))<D$(I)THENX=X+1:GOTO140
                                  ELSEY=P(X):P(X)=Z:Z=Y:X=X+1:GOTO140
145 NEXT I
160 ? : ?"**** FINISHED SORTING ****"
170 FOR C=0 TO 720 : IF P(C)<>-1 THEN ?#10, D$(P(C))
180 NEXT C : CLOSE #10 : END
```



Golden Hillock Library Suite

Background

This suite of programs was designed at Golden Hillock to improve the efficency of the existing manual system. The computer is based in the library as a school resource and setting up is only a matter of loading up the the library program at lunch time. At Golden Hillock pupil-librarians look after the registering of loan and return of books and it was early on decided that the system had to be robust and simple enough for them to use with out too much training or supervision.

The suite consists of seven programs requiring CP/M version 2.2, BASIC version 6 and preferably an operating system that can link disc sides A to C and B to D (such a system was described in Read Using 4 or 5). A printer is also considered advisable though not absolutely necessary.

The programs are in three distinct groups;

INITIALISATION.

a) INITIAL .BAS
b) FORMLIST.BAS
c) PUPWRITE.BAS

RUNNING.

d) LIBRARY .BAS
e) UPDATE .BAS

MAINTENANCE.

f) EDIT .BAS
g) FILCHECK.BAS

The Files.

The system uses two randomly accessed files on seperate discs these are :-

1) PUPMAST .DAT
2) BOOKMAST.DAT

The PUPMAST file is normally located in drive A of the computer and consists of records 56 characters long. The pupil record is accessed by entering the five figure pupil code. Each record contains the following fields:-

a)	NAME	(15 characters max)
b)	FORM	(3 characters max)
c)	BOOK ONE	(5 characters)
d)	BOOK ONE DUE	(1 character)
e)	BOOK TWO	(5 characters)
f)	BOOK TWO DUE	(1 character)
g)	BOOKS OUT	(1 character)
h)	TOTAL BOOKS BORROWED	(2 characters)

Because of the requirements of random access files the record length does not necessarily correspond with the total number of characters per field.

The bookmast file contains book records, each record is accessed by the books 5 figure code.Each BOOKMAST record contains only two fields :-

a) PUPIL NUMBER (5 characters)
b) TIMES BORROWED (2 characters)

1) LIBRARY .BAS

This is the main program of the suite, and as explained earlier it was decided to make it simple enough for unsupervised pupils to use. To enable this approach to be satisfactory most of the keyboard has been disabled. The program prompts at the beginning to change discs etc. and this sequence cannot be avoided, the program also assumes that a printer is connected so that a hard record of all loans and returns is made, each session output is headed by the date - No special format is required. The keys that remain are:-

0 - 9 (The Numbers)

The number keys are used for entering the pupil code and the book code. These numbers are entered using GET$() and after all five have been entered the program proceeds automatically (with out the need for a <RETURN>). If for some reason the number is unacceptable then an error message is produced on the screen. If a speaker is connected via the cassette port then an audible signal is also generated.

ESC

The escape button on the top left of the keyboard is used to correct entry mistakes. If pressed while entering the pupil number the operator may START AGAIN. If the escape key is pressed while entering the book number then once again the operator may start to enter the book number AGAIN. If how ever the escape key is pressed twice then the program will return to await a new pupil number.

R or r

This key signifies that a book is to be returned, it must be pressed while or before the pupils number is entered. Again a message will be flashed on the screen and an audible note sounds. Should the pupil have no books out to return then an appropriate message will appear and the program will return to await a new pupil number.

<RETURN>

The return key was added to allow the previous pupil number to be automatically retained. It was considered important as some pupils may be returning two and borrowing two books.

cont...

00000

Although not exactly a key the five zeroes are very important. By entering 00000 as either a pupil or a book number the system is automatically closed down. Like all file programs unless the files are closed then the integrity of their records cannot be guaranteed. Should - Horror of Horrors - the system for some reason crash then existing files can be preserved by typing 'CLOSE <RETURN>'.

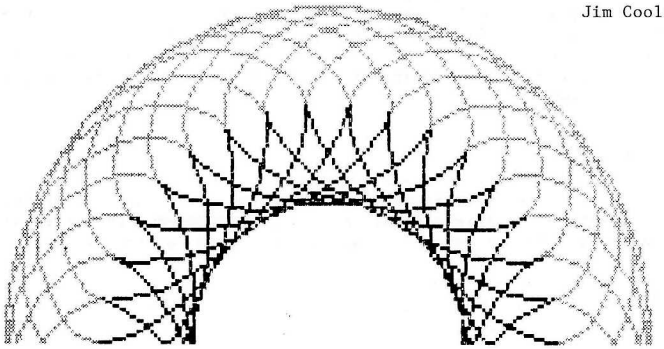
2) UPDATE .BAS

The update program is designed to be used once a week preferably after the last loan session on Friday. The program scans the pupil records and decrements the length of time allowed for each book by 1, if a book loan reaches zero then the book is overdue and an overdue notice is printed out on to the printer. The program is quite automatic and will run until it has checked each pupil record. This program may take sometime depending on how many overdue reminders are needed. As the program actually alters the pupil files it is important that it is only used once a week. If Further overdue reminders are needed then the program UPDATEP.BAS should be used, this is a passive version of UPDATE and no files are altered. Update will maintain a loan at zero if the book is continually not returned this means that it will issue reminders every week. The length of time a book is borrowed for can be altered by changing line 1120 (or there abouts) in the main LIBRARY.BAS program, and if need be update can be run once a day.

```
1120 if B$="/////" then B1$=B$:W1$="3"
                  else b2$=B$:W2$="3"
```

Change the "3"'s into however many time units you require plus one and run the update program once every time unit.

Jim Cooley



Project FLEECE

Computers in Control - Project FLEECE

In this month's Computers in Control we concentrate on a little-publicised project currently being jointly undertaken by the Ministry of Agriculture and the Ministry of Defence. Known as Project FLEECE, it is part of the MoD's attempt to bring its espionage techniques up to date. At the heart of the system is an RML Chain Network, purchased by the ministry after persuasion from their colleagues at the Department for Information Technology.

Read Using came to hear about this unusual use for the RML Network when a user reported farmyard stains on a 480Z they received back from RML after an upgrade. After a few phone calls it transpired that they had been sent the wrong 480Z. Intrigued by this use of a 480Z in Agriculture, Read Using's investigative team followed the 480Z to its rightful owner, and unearthed a surprising story. Here is their report.

The mountains of Snowdonia stand majestic above all Wales, their peaks are sharp and clear in the cloudless dawn sky. Moving down from the rocky tops, the land turns to a lush green, the dew glistening in the bright sunlight. Now and again, small black and white objects break the greenness; watch them closely and you can see them moving. These are the livelihood for many a Welsh hill farmer - Sheep. The four-legged animals provide the farmer with both wool and (dare one say it) meat, to sell to survive. An unusual setting, you would agree, for a High Technology Government research project to be taking place. But here, amongst the mountains, Disc Drives are clunking, Network Transceivers are screaming, and the latest techniques in computer espionage are being researched.

During the Falklands war the government discovered that they were having problems placing spies behind the enemy lines. The object was to record the conversations between generals in the field. They found that it was difficult for the spies to get close enough without being discovered - and disguising them as penguins was not deemed a sensible idea! After the war was over, and our victorious Ministry of Defence returned to normal they decided to set up a team of established scientists to research into easier methods of spying. Criteria listed for a good spy were:

1) The ability to remain camouflaged at all times.
2) To be portable.
3) To naturally fit in with the surrounding landscape.
4) Not to suffer from any language problems.
5) To avoid the problems of them revealing information upon capture.

Fairly early on in the scientists research they realised that humans who could satisfy all the criteria were hard to find, and certainly too good to lose. Mechanical devices were also ruled out as attracting too much attention. The un-noticeable animal seemed an obvious solution.

Animals fit into their natural landscape perfectly, no one gives them a second thought, they are always portable (although covering distances in short times can cause problems outside Africa), and there are no language problems: the enemy humans will not be able to understand what they are saying.

Project FLEECE

The next stage of the project was to investigate how the animals could be used to transmit information back to the war office. The idea they came up with was extraordinarily simple: A special microphone, implanted inside the stomach of an animal picks up the human voices of people around the animal, converts the sound into a digital signal, and passes this on to a computer attached to a network. The signals are then passed from the stations to the server, which analyses the results, stores them for future reference, and produces a print-out which can be examined by war office officials. Using a computer also enables information to be quickly forgotten. If the animal is attacked it can cry for help, and this cry is picked up by the computer (through the microphone), which then runs a program to wipe its memory banks. In fact it is this ease of erasure that helped name the project, as FLEECE stands for: Fast Loop Electrically Eraseble Computer Environment.

To try out the system Sheep are being used: Ewes for the Stations, and a Ram for the Servers. With kind help from Research Machines (who hope to gain many orders as a result of this project), the project is being field-trialled using an extended version of the Chain Network.

Early on in the project it was realised that the amount of information being passed back to the server for analysis was enormous, and it was thought that it would be more efficient if the stations cut out most of the useless information themselves. As this reduced the amount of time available to spy, a secondary network was used; which is how LAMBS (Local Area Management Based Systems) were born. Under the control of each ewe are several microphones (generally 2 or 3) placed inside lambs. The lambs dart about collecting information at very high speeds, and then pass it back to their mother system which extracts the useful information to pass back to the ram.

Software

With cooperation from Digital Research (the writers of CP/NOS and MP/M which RML use on their network) the software has been adapted to cope with this special use. The source code (previously in assembler) has been translated into a new language called MINT (Multi-Interface Network Translator), specially developed by Research Machines for this project (the first attempt was a language called BAH(sic) based on Extended BASIC version 5, but this had proved too slow). The result of this translation, followed by further development were the Crook Program Modules (CP/M), with the server running a peice of software called SHEPHERD, and the stations running FLOCK. This software provides very high transfer rates, and has automatic collision detection built in. The information is compressed as it is transmitted, a process known as Shearing, and when it arrives at the server it is spun out and woven back together to reveal the precise picture.

RML Equipment Used

As mentioned earlier, the system is designed around the RML Chain Network. As this is only a prototype system, not all the equipment is as it will be in its final version.

Project FLEECE

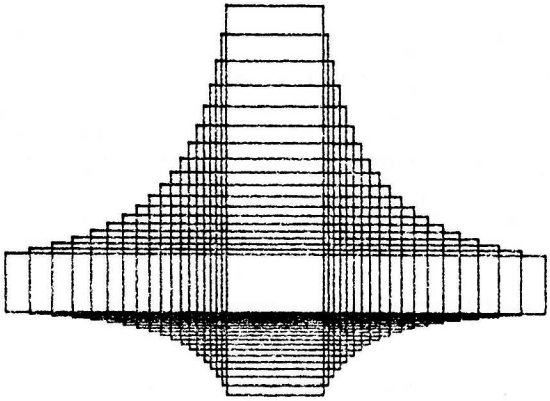
Basically, each Ewe carries a carefully camouflaged Link 480Z on its back. The 480Z circuitry is housed in a case made of Moulded Structured Wool (black or cream). The case has been designed to be extremely strong, strong enough to survive a sheep lying on it, yet at the same time soft enough for anybody touching it to think it is natural. The 480Z is connected to the microphone coder by a coaxial cable connected to the Joystick (accessory) port on the 480Z. Half way down, the cable splits into four, and attaches to four 70 ohm BNC (network style) sockets underneath the Ewe. The 480Zs are all heat powered - a rear end process providing all the power they need.

FLEECE teams work in groups of no more than 16, and the maximum distance between the first and last sheep in the chain is 300 metres - a cabling restriction (each ewe is connected onto the network in the usual way by a black coax cable).

Finalities

Project FLEECE is still in the early stages of development - many improvements still have to be made to both software and hardware (in particular the problem of still using cables for transmission remains, but RML are reluctant to alter their hardware to work with a different transmission system). Currently only sheep are used for stations and servers, but in our view it will be a lot more useful when the system is available for use with other animals. For example: Dogs, Penguins and Flying Pigs.

F. Parkes



Randomly Speaking

Have you ever needed random numbers? In BASIC there is a function provided, (which when seeded with -3 is quite good), but maybe you have tried to write your own routine? After all, what could be simpler? Answer, almost any thing.

Most home-grown random number generators are based on the theory that if you take a number, do a random number of random operations on it, you are bound to get a random result. This is not so! The only way to guarantee truly random numbers, is to apply a great deal of maths to the problem, or do what I did, look it up in a book.

Knuth deals with the subject in his own inimitable style, but the resulting algorithm is deceptively simple. There are two parts. The first, which should only be used once, simply fills a table with random(!) values. I find any odd section of the COS/ROS monitor works well! The second part actually generates the random number. The number is calculated simply by adding together the values of two bytes of the table. The two bytes are pointed to by two offsets, J and K. The random number thus formed, is also placed into the byte pointed to by K. Both J and K are then decremented and the process is repeated each time a new random number is required. It is the initial values of J and K that makes this unsophisticated algorithm so powerful. Other values could be used, but care has to be taken over their selection.

For more details see Knuth's "The Art Of Computer Programming", Vol 2

```
                ;This routine fills up Rand.table with odds and ends

0100 ED5F       Init:   LD      A, R                    ;Get random(ish) start

0102 26E0               LD      H,0E0H                  ;HL = somewhere in the
0104 6F                 LD      L,A                     ; COS/ROS monitor
0105 114201             LD      DE,Rand.table           ;DE = table
0108 013700             LD      BC,55                   ;BC = size of table
010B EDB0               LDIR                            ;Move in bytes

                ;The algorithm would fail if all the entries where even, so..

010D 216C01             LD      HL,Rand.table+42        ;Pick an entry
0110 CBC6               SET     0,(HL)                  ;Ensure its odd

0112 C9                 RET
```

continued...

```
                ;Pick a random number betweeen 0 - 255

0113 D9         Random: EXX                     ;Switch to alternative
                                                ;   registers
0114 214201             LD      HL,Rand.table   ;HL = table
0117 E5                 PUSH    HL              ;Save copy for later

0118 3A4001             LD      A,(J)           ;Get value off first pointer
011B 4F                 LD      C,A
011C 0600               LD      B,0             ;BC = first offset
011E 09                 ADD     HL,BC           ;Add to start of table
011F 7E                 LD      A,(HL)          ;Get first value
0120 08                 EX      AF,AF'          ;Save

0121 E1                 POP     HL              ;HL = Start of table
0122 3A4101             LD      A,(K)           ;Get value of second pointer
0125 4F                 LD      C,A             ;BC = second offset
0126 09                 ADD     HL,BC           ;Add to start of table
0127 7E                 LD      A,(HL)          ;Get second value
0128 47                 LD      B,A             ;Into B
0129 08                 EX      AF,AF           ;Get first value
012A 80                 ADD     B               ;Add together
                                                ;A = random number

012B 77                 LD      (HL),A          ;And store at table+K
012C 08                 EX      AF,AF           ;Save result  byte

012D 214101             LD      HL,K            ;Point to K
0130 35                 DEC     (HL)            ;Dec K
0131 2002               JR      NZ,R1
0133 3637               LD      (HL),55         ;If 0 then K=55

0135 214001     R1:     LD      HL,J            ;Point to J
0138 35                 DEC     (HL)            ;Dec J
0139 2002               JR      NZ,R2
013B 3637               LD      (HL),55         ;If 0 then J=55
013D            R2:
013D 08                 EX      AF,AF'          ;Get result

013E D9                 EXX                     ;Back to normal regs

013F C9                 RET


0140 18         J:      DEFB    24              ;Initial values
0141 37         K:      DEFB    55

0142            Rand.table:
0142                    DEFS    55              ;Must be at least as big K
```

I use the following example to show one method of adding USR routines to BASIC. The machine code routine (bubble) sorts a string of characters into order. Details of the string are passed to the routine using the "VARADR" and "USR" functions. It is interesting to also sort the string using MID$ (BASIC version 6 is very handy for this) and then compare the speed!

The function VARADR(A$) gives the (decimal) address of a four byte "description block" containing the number of elements in the string, zero, and the (two byte) address of the string in memory.

```
                ;SORT program for "USR" and "VARADR"

0100                    ORG 100H

0109 =                  USRVF EQU 109H
0115 =                  E.ERR EQU 115H
                ;-----

0100 CD0901             CALL USRVF      ;Get value
0103 D5                 PUSH DE
0104 DDE1               POP IX          ;Copy into IX
0106 DD7E01             LD A,(IX+1)     ;Second byte is zero
0109 FE00               CP 0
010B C21501             JP NZ,E.ERR     ;Incorrect call made

010E DD7E00             LD A,(IX+0)     ;No. of elements
0111 FE02               CP 2
0113 381F               JR C,FIN        ;Nothing to do
                ;-----

0115 DD4600     TOP:    LD B,(IX+0)     ;No. of elements
0118 05                 DEC B           ;Pairs
0119 DD6E02             LD L,(IX+2)
011C DD6603             LD H,(IX+3)     ;HL points to data list
011F 0E00               LD C,0          ;Clear flag

0121 7E         CPR:    LD A,(HL)
0122 23                 INC HL
0123 BE                 CP (HL)
0124 3809               JR C,CNT        ;Correct order
0126 2807               JR Z,CNT        ;Equal

0128 56                 LD D,(HL)       ;Swap elements
0129 77                 LD (HL),A
012A 2B                 DEC HL
012B 72                 LD (HL),D
012C 23                 INC HL
012D 0E01               LD C,1          ;Set flag
012F 10F0       CNT:    DJNZ CPR        ;Next pair
0131 0D                 DEC C
0132 28E1               JR Z,TOP        ;Flag set - do another pass

0134 C9         FIN:    RET             ;Sort complete - return to BASIC
```

continued...

Note the use of relative jumps to produce a routine in Position Independent Code that will run at any address. The machine code instruction JP E.ERR (the value is given in the BASIC release note) returns to BASIC printing an "External Error" message.

The HEX file produced when the routine is assembled is:-

```
:10010000CD0901D5DDE1DD7E01FE00C21501DD7EF8
:1001100000FE02381FDD460005DD6E02DD66030EBF
:10012000007E23BE3809280756772B72230E011054
:05013000F00D28E1C9FB
:0000000000
```

This file can be read by TXED and edited into DATA statements for the BASIC program. (Note that you will need to delete the first nine and last two characters from each line of the HEX listing.)

An edited form becomes:-

```
300 DATA CD0901D5DDE1DD7E01FE00C21501DD7E
310 DATA 00FE02381FDD460005DD6E02DD66030E
320 DATA 007E23BE3809280756772B72230E0110
330 DATA F00D28E1C9
```

and a BASIC loader program could be:-

```
10 REM General USR routine for BASIC V5/6
20 REM SORT used as example
30 REM
100 CLEAR 200,,256:REM 256 bytes for USR routine
110 CA=PEEK(&11C)+256*PEEK(&11D)+1:REM Start of cache
120 POKE &107,CA-INT(CA/256)*256
130 POKE &108,INT(CA/256):REM Point USR vector
140 S=CA
150 FOR I=1 TO 4:REM 4 data items
160 READ U$:L=LEN(U$)
170 FOR J=1 TO L STEP 2
180 POKE S,VAL("&"+MID$(U$,J,2)):S=S+1
190 NEXT J
200 NEXT I
300 DATA CD0901D5DDE1DD7E01FE00C21501DD7E
310 DATA 00FE02381FDD460005DD6E02DD66030E
320 DATA 007E23BE3809280756772B72230E0110
330 DATA F00D28E1C9
1000 REM Now add your program e.g.
1010 INPUT A$
1020 A=USR(VARADR(A$))
1030 PRINT A$
1040 END
```

Note 1) The syntax of CLEAR 200,,256.
2) The lines 110, 120 and 130 enable the program to run in any memory configuration.
3) We can use the instruction CALL USRVT (USRVT EQU 10CH - see the BASIC manual) to transfer a value from our routine to what would be the variable A above.

Barry Mawer

Disk Reset Function for TCL Pascal

(This article was written: Paul R Spurgeon and originally appeared in "Chiltern Computing" in February 1984).

Why hasn't anyone spotted this omission before? It is not possible to cause a 'warm boot' from within RML (TCL) Pascal. The procedure call that is missing is the equivalent to BASIC's "reset" command. The result of this is that any program that gives you the chance to change disks before sending output to a file can give you instant BDOS errors and throw you out of the program! Fortunately Pro-Pascal includes the procedure "dreset", and I expect the next release of RML's Pascal to include this too. Meanwhile, the following code may be included in a program after compilation, linking and locating:

```
; DRESET... RML (TCL) Pascal disk reset
; Copyright Paul R Spurgeon (13 December 1983)

*I C:USER       ; Pascal organisation file

; PROCEDURE dreset;

; Performs a reset of the disk system, so file access may continue
; after a disk change.  The default drive is re-established as it was
; before the call.

BDOS    = 5
LOGGED  = 19H
RESET   = 0DH
LOGIN   = 0EH

EXT0:   POP     IY              ; Keep return address in IY
        LD      BC, 0
        LD      C, LOGGED       ; Get currently logged drive
        CALL    BDOS
        PUSH    AF              ; and save it

        LD      C, RESET        ; Do the deed
        CALL    BDOS

        POP     AF              ; Recover drive and log back
        LD      D, 0            ; onto it
        LD      E, A
        LD      C, LOGIN
        CALL    BDOS
        JP      (IY)            ; Return to the return address
;----

EXT1:
EXT2:
EXT3:           ; Since there are no other external routines
EXT4:           ; all of these labels fall through to print
EXT5:           ; an error message and reboot if they are used.
EXT6:
EXT7:
EXT8:
```

continued...

```
EXT9:
EXT10:
EXT11:
EXT12:
EXT13:
EXT14:
EXT15:  LD      HL, EMSG        ; Print error message
        EMT     MSG
        JP      BOOT            ; Reboot

EMSG:   13,'External vector error',13,-1

MSG     = 23
BOOT    = 0
;----

EOP     = 26

PASEND: DEFB    EOP

END
```

The normal technique for inclusion of machine code should be followed:

```
Program user(assembler);
(*  put assembler switch in main program header  *)
    :
PROCEDURE dreset; EXTERN 0;
(*  declare the dreset procedure as a call to EXTERN 0  *)
    :
    :
BEGIN
    :
  dreset;       (*  call as normal within program  *)
    :
END.
```

After compiling, linking and locating, an organisation file (USER.ZSM) will be produced. This is included in the assembly source code, which is assembled in the normal way using ZASM. Then use DDT to overlay the DRESET.HEX file on top of the USER.COM file:

```
A>DDT USER.COM

-IDRESET.HEX

-R

-C

A>SAVE xxx USER.COM
```

where xxx is the size in 256 byte pages of the resulting file to be saved.

Easier to do than to read about; but I for one look forward to a new version including this and other features.

Paul R Spurgeon
17-1-84

380Z Joysticks.
****

This joystick is an adaption of an article printed in 'Computers in Schools' December 1981 by Gavin Bell.

Mr Bell describes how a simple 555 timer chip may be connected to the user I/O port so that it will return a value between 0 and 255 depending on the value of the capacitor/resistor timing network R1 and C1.

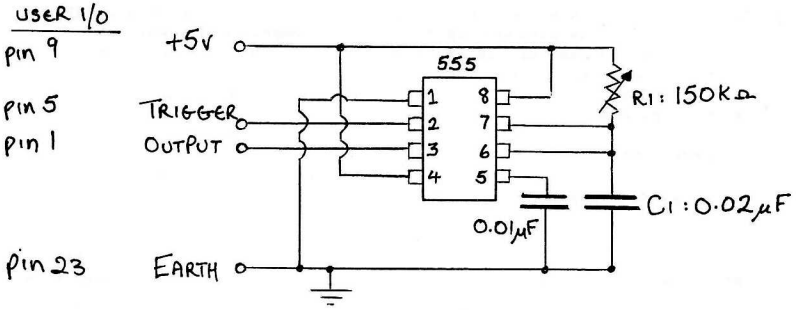
The 555 receives a trigger pulse from the 380Z and then sends back a pulse, the duration of which depends upon the setting of the variable resistor. A software counter measures the duration and returns a value between 0 and 255.

This system is very simple and with a machine code subroutine can be swift enough for this type of application. The range of resistance can easily be altered to suit the resistance sensor, it is the TIME CONSTANT that is important.

TIME CONSTANT = RESISTANCE (Ohms) * CAPACITANCE (Farads)

So all that is needed is to alter the capacitor to maintain the same time constant of 0.003.

The following circuit may be adapted for a joystick by either using two 555s or a single 556. Upto 8 channels may be used but is useful to use some of the user I/O for fire buttons and other switches. Any resistive device may be attached to the circuit with the appropriate changes to maintain the time constant. Suitable devices are Light Dependent Resistors, Thermistors and any form of variable resistor.



Single Channel Circuit.

cont...

ANALOG - source and object code.

```
21  FF  FB     INIT     :LD HL,0FBFFH    ;HL=USER PORT
C1                       POP BC          ;B=0 RESET COUNTER
79                       LD A,C          ;C=CHANNEL NUMBER
2F                       CPL             ;A=INVERSE CHANNEL

36  FF         START    :LD(HL),0FFH     ;SEND START
77                       LD(HL),A        ;PULSE TO 555
36  FF                   LD(HL),0FFH     ;TIMER CHIP
7E             COUNT    :LD A,(HL)       ;WATCH FOR PULSE FROM 555
A1                       AND C           ;MASK UNWANTED BITS
28  02                   JR Z RETURN     ;PULSE FOUND
10  FA                   DJNZ            ;PULSE NOT FOUND

78             RETURN   :LD A,B          ; INVERT COUNT
2F                       CPL             ;   AND PASS
47                       LD B,A          ;   BACK IN
AF                       XOR A           ;    REG AB
E1                       POP HL          ;RETURN ADDRESS
CD  0F  01               CALL SAVVAR     ;RETURN VALUE TO
C9                       RET             ;BASIC VARIABLE
```

This routine is relocatable apart from assuming the address of SAVVAR. This code may be poked onto memory using the following BASIC program.

```
10 HM=PEEK(6)+256*PEEK(7)
20 ZZ=HM-30
30 FOR I=ZZ TO ZZ+25
40   READ J:POKE I,J
50 NEXT I
60 DATA 33,255,251,193,121,47,54,255,119,
        54,255,126,161,40,2,16,250,120,47,
        71,175,255,205,15,1,201
70 REM
80 REM CONTINUE WITH PROGRAM
90 REM CHANNELS ARE 1,2,4,8,16,32,64,128
100 REM
110 CALL ZZ,VARADR(X),1
120 REM ZZ is the address of routine
130 REM      VARADR(X) is the location of
140 REM                1 is the channel.
150 PRINT X
160 GOTO 110
```

Connections to USER I/O:

+ 5v	PIN 9
earth	PIN 23
trigger	PIN 5
output	PIN 1

Jim Cooley

Digitimer D204 RAM Disk

A RAM Disk (often called a 'Silicon Disk') provides you with an extra place to store programs and data allowing very fast access. In use the RAM Disk is indistinguishable from a disk drive except that it makes none of the usual grinding clunks and any information stored on it is lost when the computer is turned off. Up to 254K of programs can be stored on the RAM Disk.

The Digitimer RAM Disk comes as a single printed circuit board which slots into your 380Z like the rest of the boards. Also supplied is a disk containing a few handy programs and some text files that comprise the entire documentation. One of the programs supplied called INSTALL must be run whenever you boot up the computer. This installs the RAM Disk by making CP/M recognise that it has another drive connected.

After running INSTALL you will have an extra drive as well as your usual disk drives A:, B:, etc. The new drive can be named anything between E: and P: by giving INSTALL an argument. For the review we used the RAM Disk as drive I: which is the default. You can STAT this drive, log onto it, PIP programs to and from it and do anything else that you can do with a normal disk drive. To take advantage of the RAM Disk you have to copy those programs and files that you are going to use most frequently onto drive I: where they can be loaded or accessed very quickly. One of the programs that is supplied speedily copies all of one side of a disk onto drive I:. This takes 17 seconds for a mini disk and 27 seconds for a full (8") disk. The idea is that at the start of a session you could copy everything that you need into the RAM Disk and then work at lightning speed while logged onto drive I:. The RAM Disk does not lose any information if you press the reset button and restart the machine.

Speed

This Lightning Speed is the main thing that the RAM Disk offers. However, the speed gained depends very much on your application. Some rough timings are given below comparing the RAM Disk with a single density 8" disk drive.

RAM Disk	8" Disk	Task
0.2 sec	5 sec	Loading BASICS - an 13K 'COM' file
1 sec	15 sec	Loading a 40K 'COM' file
1m 12 sec	1m 30 sec	Loading a 17K BASIC program (BASIC V5.0L)
9 sec	14 sec	Writing 1000 numbers serially from BASIC
6 sec	12 sec	Reading " " " " "
6 sec	20 sec	Assembling an 8K program using ZASM
2m 03 sec	3m 11 sec	Compiling a 44K Integer BASIC program
15 sec	40 sec	Searching through a 110K DATASTAR file

The most breathtaking speed is found when loading things like BASIC or TXED from CP/M command level. This is almost instantaneous. Most other applications, however, ran one and a half times as fast using the RAM Disk. Loading programs from BASIC was not much improved. Since this is what most of us spend most of our time doing, the RAM Disk loses some of its appeal.

Usefulness

Programs like WORDSTAR which are too big to fit comfortably into memory have to store some routines on disk. These routines (or 'overlays' as they are called) are only loaded and run when they are needed. Many of WORDSTAR's commands and help information are stored like this - typing CTRL Y to delete a line results in a few seconds of clunking before the line is deleted. If the RAM Disk is used to store the overlay files then the waiting is over.

DATASTAR, MICROQUERY and similar data storage and retrieval programs are fast if the database files are stored on the RAM Disk. Compilers that have to write intermediate files and search through libraries can take advantage of the RAM Disk. HELP programs can provide instant help if the information libraries are kept on the RAM Disk. With a normal (single density) mini disk 380Z you have only 72K bytes of space per side of a disk. Having an extra drive with 254K bytes of space could be very useful in some applications.

Documentation

This was copious and comprehensive. While it could be a little friendlier it is quite adequate for people with experience of CP/M (who therefore do not expect friendly manuals). Circuit diagrams and details of how to program the board are given for people with specialist applications. The assembly language sources of all the RAM Disk software are given. These are reasonably commented and would enable the experienced assembly language programmer to use the RAM Disk for a special application. With some fiddling with the software you could chain two or more RAM Disks together to give 0.5 Megabyte or more of space.

Price

The RAM Disk costs £ 450 - a mere £ 50 more than a BBC model B. The board is a good quality double sided board, with socketed chips, etc., but it is still very expensive. Few schools will be buying one at this price - to tap the educational market the price would have to be at most £ 200. The extra volume of sales at the lower price might well justify the reduction. RML have promised a 190K RAM Disk for the 480Z some time this year. It might be a bit cheaper, but of course won't help your 380Z.

Other musings

If it were possible to link the INSTALL software into MP/M then it should be possible to use the RAM Disk in a network server. The large amount of system software that is needed on the server could then be kept on the RAM Disk where it could be instantly accessed. This should speed up the network fantastically (if it were to work!)

The RAM Disk was tested (and worked!) on COS 3.4 and COS 4.0 380Z machines with mini and 8" single density disk drives. Software for double density drives is supplied in the package but was not tested.

Ed.

To Print Low Resolution Graphics Pictures with an Epson MX-80

Yet another gratefully received offering from the golden hillocks of Brum - this program enables you to print low resolution graphics pictures on your super hi-tek Epson MX80 which apparently cannot normally do this. James T Cooley, the well known computer scene commentator, was quote "appalled" to find that this printer can only list things and print high resolution graphics pictures but not do anything useful. I think that printer manufacturers go to very great lengths to provide interesting programming projects for pupils and this service should be applauded. Ouch! I've bitten my tongue.

```
10 REM LOW RESOLUTION SCREEN DUMP FOR EPSON MX-80 TYPE III
11 REM  Written by P. Desmond
12 REM  Golden Hillock School
13 REM  Sparkhill
14 REM  Birmingham
15 REM
16 REM  Performs a screen dump of Low Res. graphics
17 REM  characters in the range 128-255 (ASCII)
18 REM  Uses 40-character screen.
19 REM  Characters in the range 3-31 are not available yet.
20 CLEAR 1000
30 DIM CH(63,2),U(78,2)
40 REM READ GRAPHICS CHARACTER SET FROM DATA
50 FOR I=1 TO 63
60 READ CH(I,1),CH(I,2)
70 NEXT I
80 PRINTER 4,6
90 PUT#2,27,"A",5,27,"E"
100 PRINT"OMIT BLANK LINES AT TOP OF SCREEN (Y or N)";
110 BL=GET():IF BL>90 THEN BL=BL-26
115 IF BL<>89 AND BL<>78 THEN 110
116 PUT BL,13
120 FOR Y=57 TO 0 STEP -3
130 SP$=""
140 FOR X=0 TO 78 STEP 2
150 C=POINTS(X,Y)
160 IF C=32 OR C=128 OR C=192 THEN SP$=SP$+" ":GOTO 280
170 IF C<31 THEN ?"UNRECKOGNISABLE CHARACTERS":STOP
180 CF=-1:PUT#2,SP$:SP$=""
190 IF C<127 THEN PUT#2,C:GOTO 280
200 IF C<192 THEN D$="K":N=6:C1=C-128 ELSE D$="L":N=12:C1=C-192
210 PUT#2,27,D$,N,0
220 FOR J=1 TO 2
230 FOR K=1 TO N/2
240 PUT#2,CH(C1,J)
250 NEXT K
260 IF CH(C1,J)/2>INT(CH(C1,J)/2) THEN U(X,J)=N/6
270 NEXT J
280 NEXT X
```

SP 0/10

cont...

```
290 IF CF=0 AND BL=89 THEN 460
300 LPRINT
305 IF CF=0 THEN 450
310 FOR X1=0 TO 78 STEP 2
320 IF U(X1,1)=0 AND U(X1,2)=0 THEN PUT#2,32:GOTO 440
330 FOR J=1 TO 2
340 IF U(X1,J)=1 THEN N(J)=24:D$="K":N=6
350 IF U(X1,J)=2 THEN N(J)=24:D$="L":N=12
360 IF U(X1,J)=0 THEN N(J)=0
370 U(X1,J)=0
380 NEXT J
390 PUT#2,27,D$,N,0
400 FOR J=1 TO 2
410 FOR K=1 TO N/2
420 PUT#2,N(J)
430 NEXT K:NEXT J
440 NEXT X1
450 LPRINT
460 NEXT Y
470 DATA 224,0
480 DATA 0,224
490 DATA 224,224
500 DATA 28,0
510 DATA 252,0
520 DATA 28,224
530 DATA 252,224
540 DATA 0,28
550 DATA 224,28
560 DATA 0,252
570 DATA 224,252
580 DATA 28,28
590 DATA 252,28
600 DATA 28,252
610 DATA 252,252
620 DATA 3,0
630 DATA 227,0
640 DATA 3,224
650 DATA 227,224
660 DATA 31,0
670 DATA 255,0
680 DATA 31,224
690 DATA 255,224
700 DATA 3,28
710 DATA 227,28
720 DATA 3,252
730 DATA 227,252
740 DATA 31,28
750 DATA 255,28
760 DATA 31,252
770 DATA 255,252
780 DATA 0,3
790 DATA 224,3
800 DATA 0,227
810 DATA 224,227
820 DATA 28,3
830 DATA 252,3
840 DATA 28,227
850 DATA 252,227
860 DATA 0,31
870 DATA 224,31
880 DATA 0,255
890 DATA 224,255
900 DATA 28,31
910 DATA 252,31
920 DATA 28,255
930 DATA 252,255
940 DATA 3,3
950 DATA 227,3
960 DATA 3,227
970 DATA 227,227
980 DATA 31,3
990 DATA 255,3
1000 DATA 31,227
1010 DATA 255,227
1020 DATA 3,31
1030 DATA 227,31
1040 DATA 3,255
1050 DATA 227,255
1060 DATA 31,31
1070 DATA 255,31
1080 DATA 31,255
1090 DATA 255,255
1100 END
```

Extended BASIC version 6 - Random Access Files

In Read Using 7, I wrote about some of the features provided in RML's new BASIC 6 interpreter. In that article, I promised that I would discuss the random access files facilities in a separate article in Read Using 8. Here is that article.

At this point, I could start with another quotation from an RML document, saying that they were going to release a Random Access Files BASIC very soon. There is a particularly good one in the HRG manuals released between 1979 and 1983. However, I think we've embarrassed RML enough over this, and we should be concentrating on something else now (SBAS?, Network performance?, Silicon Disc?....). Whilst on the subject of grovelling to RML (will I get my network upgrade done any quicker?) I actually think that the Random Access part of BASIC 6 is very good - it actually seems to let me do most things I want to do, without having to do too much messing about myself, my only worry is that it does too much, and hence is too complicated.

Why do I want to use Random Access Files?

Firstly, perhaps I ought to try and explain what a Random Access File is. You would probably do better to read a published book about it (I found 'A Practical Approach to Computing' by Arms, Baker and Pengelly quite good on this), but here is my explanation anyway. Files are normally stored on a disc in a continuous fashion, - the first thing you write to it goes at the beginning, the next bit of data goes immediately after that, and so on. This is fine when writing the data, but when you come to read it back you may find that the bit of data you want to read is actually the 32nd piece. This means that you have to read, and throw away, the first 31 pieces of data, before you can read the piece you want. If you next want to read the 25th piece, you have to start again at the beginning of the file and skip through until you reach that piece. Doing this can be a time consuming task, especially if there are over a hundred pieces of data. With a Random Access File, you can save an enormous amount of time - you ask it for the 32nd piece, and it automatically positions itself at the beginning of that piece of data, and then reads it - it hasn't had to read the other 31 pieces first. Then, when you want to read the 25th piece, it just positions itself at the beginning of that data, and reads it, without having to start again at the beginning. Reading data this way obviously speeds up the operation enormously.

Another, even more useful, facility provided by Random Access Files, is the ability to be able to update pieces of data in the middle of a file. Using serial files, you have to read through the file, writing it out to another file at the same time, until you get to the piece of data you wanted to modify. You can then alter this, and write the new version out. After this, you must then read (and write) through the rest of the file in order to keep that data. Updating data in this way is not really on! With Random Access Files, you can tell it which piece of data you want to alter, and it will move to the correct place in the file, and read it. You can then alter it, and write it back again to the same place, without having to do any copying of the rest of the file - a vast improvement.

When using a random access file, each of these pieces of data is called a **record**, and the number that you use to access a particular record is called the **record number**.

Sounds Good - What are the drawbacks?

The major problem is space - a random access file tends to use up more space on the disc. With a serial file, each piece of data can follow the previous one exactly, only one character is needed to say where the end of one piece finishes and another piece starts. In order to be able to move about the file so quickly, the Random Access File needs to know how big each piece of data is. It does this by making them all the same length. The drawback of this is are that the size of the data you can put in is restricted - if you try and fit too much in it will complain, and if you don't put very much in, then a lot of very valuable space will be wasted. Another problem is that you have to say what the maximum size of the data you want to put in the file is, before you have even written anything to it - if you get it wrong then you've got problems.

Using RML's RAF

With RML's random access files system, you must state the size of the records when you open the file. The command:

```
RANDOM #35,"MYFILE",WRITE,102
```

is used to say that I want to use a random access file called MYFILE, on channel number 35 (channel numbers can be between 20 and 127), I want to be able to read from, and write to, it (this is what the WRITE means - if you put READ there instead then it stops you writing to the file), and that I want to use a record length of 102 characters.

To read a record from the file you would use the READ command. A program to read the 32nd record followed by the 25th record would look like this:

```
10 RANDOM #35,"MYFILE",WRITE,102
20 READ #35 @ 32, A$
30 READ #35 @ 25, B$
40 CLOSE #35
```

The @ sign in lines 20 and 30 indicate that the number following it is a record number. In this example the records are read into the variables A$ and B$. The CLOSE command is used to close the random file.

To alter a record this program could be used:

```
10 RANDOM #35,"MYFILE",WRITE,102
20 READ #35 @ 32, A$
30 WRITE #35 @ 32, A$+"CORRECTED"
40 CLOSE #35
```

The WRITE command works in the same way as the READ command.

continued...

Reviewing RML's RAF

This review does not attempt to describe the random access commands in detail - that's what manuals are for! (even if you can't understand them).

Opening Files

Files are always opened using the RANDOM command. If the file does not exist on the disc, then it creates it for you. When it creates a file it writes a header to the file which contains information such as the record length for the file. This means that next time you want to use it, you don't have to specify what record length you actually want to use - it automatically takes it from the header.

I actually found the fact that it created the file if it did not exist slightly annoying, as if you spell the name wrong, you end up creating a file, instead of it telling you it doesn't exist - I would have preferred to have had two separate commands, or an option on RANDOM. One way round this problem is not to specify a record length if you know you are using an existing file (the last two parameters are optional). Then, if the file does not exist, it will complain, telling you that you must specify a record length.

Calculating the record length you wish to use can be a tricky business, and I usually found it useful to write out first what I was going to put in a record, and the maximum number of characters I was going to allow for each item. In RML's random access files, the data is stored in an internal format. This means that numbers always take 5 characters, and strings take 2 characters more than the length of the string. Strings and numerics can be mixed in the same record, and each record need not have them in the same order - a very useful feature.

Reading and Writing

The READ and WRITE commands are very similar, and therefore can be discussed together. These commands are quite flexible. Normally one would use them by specifying a record number, and the data to read/write. It is however, possible to omit either, or both, of those two parts. Omitting the data part just moves you to the start of a particular record. Omitting the record number has more interesting effects: the data is read/written from the current position in the file. This allows you, for example, to read half of one particular record at one time, and later come back and read another part. Perhaps an example would clarify here:

The program snippet:

```
READ #76 @ 12, N, A$, Q$
```

could be rewritten:

```
READ #76 @ 12
READ #76, N
READ #76, A$, Q$
```

Now that may not seem very useful, it just involves more typing, but there are times when it is a helpful program aid. For Example: the first data item in the record (N) could indicate how many more data items there were in the record. This could be useful for an examination system, when you

want to know how many subjects a person is taking, and what they are. The example above could be more realistically expanded to:

```
READ #76 @ PN            'Move to record for the pupils number (PN)
READ #76, NS             'Read number of subjects
FOR I = 1 TO NS
  READ #76, S$(I)        'Read subject name
NEXT I
```

Now do you see the use?

When reading/writing sequentially, in the fashion described above, you are allowed to stretch across more than one record, but BASIC 6 carefully makes sure that individual data items are never split across records, if it won't fit in the current record it is placed at the beginning of the next.

Other bits and pieces

In addition to the READ and WRITE commands, there are a number of other functions, some of which provide useful facilites, particularly when reading/writing in a sequential fashion. The most useful of these functions is the TYP function. This function returns the ASCII value of the next character in the file. This value can be used to determine whether there is another data item in the record, and if there is, whether it is string or numeric. This prevents you from getting type mismatch errors when you try to read it - it's a shame it only works with random access files. An example is given in the manual, so I won't repeat it here. The example program (MEMBER) on the BASIC distribution disc also uses the TYP function to detect whether a record has been deleted (a delete record facility is a sad omission from BASIC 6).

The other functions FPOS, FLEN, RPOS, and RLEN return values indicating, your current record number, the last written record in the file, the record length, and your position within the record. Of these only FLEN is really useful, as this can be used to help validate a record number, before using it in a random access command.

Conclusions

The Random Access System provided on BASIC 6 by RML is generally good. It is not very friendly to beginners - storing the data in internal format does complicate matters, particularly when calculating the record length. For a more experienced user however, it is probably ideal, as the serial access facilities help them to manipulate the data it whatever way is required.

continued...

Comparisons of Random Access Systems

Random Access Files Version 3 by Software Production Associates

These RAF facilities have been available now for a couple of years (version 2 was released 6 months before that) and are in use in many schools and colleges (those that were fed up of waiting for RML).

The random access system offered by SPA has a number of obvious disadvantages over the system provided by RML. The biggest disadvantage is that you can only write strings to the file, and then only one (giving a maximum record length of 255 characters, as opposed to the 65535 offerred by RML).

Only being able to write strings is a major drawback compared to RMLs system. The standard Microsoft random access system has the same problem. If you want to write a number you have to first convert it to a number (using STR$), and then to read it back you have to reconvert it back into a number (using VAL). To further complicate matters, only one string can be written. This means that if you want to write more than one to the same record you have to join them together using the string concatination operator +. Also, you will have to make sure each string is of a fixed length, so that it can be separated again when read back. SPA do provide you with two functions PAD$ and UNPAD$ which allow you to create strings of a fixed length for writing, and also strip off unwanted spaces after it has been read back. This way of doing things is a bit messy (but you could always hide it in an SBAS procedure!)

The disadvantages given above are not insurmountable, it just makes programming more complicated.

The SPA system does have some advantages over RML's. The commands for creating a new random file or opening an existing one are different - something I complained was wrong with the RML system. SPA also provides an initialise command, which initialises a random access files to sensible values - if you don't you are liable to read rubbish from records which have not been written. In RMLs version you have to specifically write a value to each record, which obviously takes longer (and is more complicated) than a specific command to do it for you.

Another advantage of SPA's system is the use of a special buffer. Records to be written are placed in this buffer, rather than being immediately written to the disc. When the buffer becomes full, all the records are written at once. As they are placed in the buffer they are sorted into an order which optimises the number of disc accesses required (or so the manual claims), so that the actual writing process is speeded up. Speed is also increased on reading, as if the record to be read is in the buffer already, it is taken from there, rather than from the disc, and thus the operation is instantaneous. The size of the buffer used can be altered by the user, and the user can also decide not to use it at all if it is more suitable not to.

In some places the SPA system is friendlier. One example of this is that you don't need to be bothered with channel numbers if you don't want to. With SPAs system you can always refer to the file by name rather than by number. Another friendly facility is the RFILES command. This command

prints out details about the random access files you currently have open - useful when debugging. It is a shame it doesn't cover serial files as well.

SPAs system does not use an internal format. This means that files created in a different language can be read (RMLs system can only read RML files). In particular BASIC serial files - it is usually quicker than using repeated GET statements if you want to use each character in the file individually.

SPA's random Access Files work with CP/M 1.4 (as well as 2.2) - RML restrict theirs to CP/M 2.2 . In addition to saving on the upgrade fee, this also provides an extra 1K of program space. On the subject of space it is useful to note that BASIC 5 with SPA's RAF is only 16K, whereas RML's is 20K - a saving of 4 (extremely valuable) K.

Conclusion

In general the SPA system is not as useful for the experienced user as the RML system - there are a number of restrictions in the reading and writing of records. Also, SPA's does not offer the facility of reading and writing only parts of a record. However, SPAs does win in the friendliness of the supporting functions (initialising files, use of name rather than channel number) and also has the advantage of some software having already been written in it. Furthermore you can use it with SBAS - that has to be worth it on its own. Why not buy both?

(A speed comparison between RML's and SPA's system is given at the end of this article.)

BBC System

Before doing any comparrison, I ought to confess to knowing very little about the way the BBC system works - my experience is gained from reading the manual, and a short time experimenting. It is possible I have overlooked some extremely useful command, if I have I suggest somebody writes to READ USING and complains!

The BBC Micro does not offer a proper Random Access system. What it does offer is the ability to move a pointer to any position in the file and then read or write to it as one would to a serial file. This offers enormous flexibility for the programmer to do exactly what he wants, but can make life hard for the beginner. On the BBC system the data in the file is stored in internal format, this means that to use the system usefully you have to make sure that you always start reading at the beginning of a data item as otherwise the data read will be confused - it is like trying to use an operating system rather than a high level language. It is, of course, possible to build up a set of procedures to provide proper random access facilities, and if you wish to use random access on a BBC this is probably the best way of going about it.

In conclusion, the two systems are not really comparable. The BBC system is very crude, and is only really useable if you know what you are doing - I'd go for RML every time!

continued...

Timing Comparison

To try and assess the relevant merits of SPA's and RML's random access files systems the following 3 benchmarks were used.

1) Read 50 records randomly.
2) Write 50 records randomly.
3) Update 50 records randomly.

For each of the tests the same set of random numbers were used to ensure fairness. The first part of each of the programs read the random numbers into an array, so that part is not reproduced here.

The tests were performed on a 380Z MDS-2 (single density) using CP/M 2.2.

All timings are approximate and given in seconds.

1) Reading Records

RML version

```
100 RANDOM #20,"RANDFILE",WRITE,100
110 FOR I=0 TO 49
120 READ #20   RN(I),REC$
130 NEXT I
140 CLOSE #20
```

Time: 35.5 seconds

SPA version

```
100 ROPEN #20,"RANDFILE",100,1000
110 FOR I=0 TO 49
120 RREAD #20, REC$, RN(I)
130 NEXT I
140 RCLOSE #20
```

34.3 seconds

2) Writing Records

For this test it was found that there was a considerable difference in the performance of SPA's system, depending on the size of buffer that was being used. Therefore timings are given for buffer sizes of 512 bytes (the default), 1000 bytes (a more likely size), and 5000 bytes (in this case all the records fitted in).

```
100 RANDOM #20,"RANDFILE",WRITE,100
110 FOR I=0 TO 49
120 WRITE #20   RN(I),"This is
    record"+STR$(I)
130 NEXT I
140 CLOSE #20
```

Time: 62.0 seconds

```
100 ROPEN #20,"RANDFILE",100,1000
110 FOR I=0 TO 49
120 RWRITE #20, "This is record"+
    STR$(I), RN(I)
130 NEXT I
140 RCLOSE #20
```

Buffer Size	Time
512	56.3 seconds
1000	44.0 seconds
5000	16.0 seconds

Random Access Files

3) Updating Records

Again three different buffer sizes were used for SPA's benchmarks.

```
100 RANDOM #20,"RANDFILE",WRITE,100
110 FOR I=0 TO 49
120 READ #20   RN(I),REC$
125 WRITE #20   RN(I),
    "CORRECTED"+REC$
130 NEXT I
140 CLOSE #20
```

Time: 76.5 seconds

```
100 ROPEN #20,"RANDFILE",100,1000
110 FOR I=0 TO 49
120 RREAD #20, REC$, RN(I)
120 RWRITE #20, "CORRECTED"+REC$,
     RN(I)
130 NEXT I
140 RCLOSE #20
```

Buffer Size	Time
512	99.3 seconds
1000	84.0 seconds
5000	53.1 seconds

Deductions

So what can we deduce from the above figures?

It is fairly safe to say that there is no difference in reading records - the operations probably spend more time waiting for the discs than anything else. We can call that one a draw.

With writing it is obvious that the buffer makes an enormous amount of difference - 16 seconds for writing 50 records is substantially faster than 56 seconds. This is presumably because the records are sorted into a sensible order before writing, to avoid moving backwards and forwards across the disc. However, 5K is a lot of space to reserve for a buffer, you may not be able to afford that much - if you can use a larger buffer it is obviously worth it though. SPA wins this one.

On the updating benchmark it is noticeable that even with a buffer size of 5000 bytes SPA's is not substantially faster than RML's - it takes 2/3's the time. As the buffer becomes smaller - SPA's actually drops behind. This one is probably a draw, with a slight advantage to RML, as it is unlikely that such a large buffer would be used in normal operation.

Overall, there is not mush to choose between the two systems - they spend most of their time waiting for CP/M to access the disc.

G. Dale

HELP!

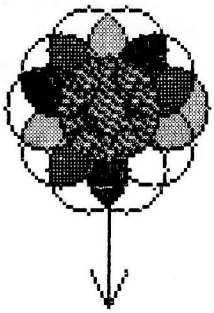
Any assistance that you can give to these people will be greatly appreciated.

Perhaps you could give some directions for the lunchtime computer club that I run. Enthusiasm has waned somewhat and I want to get things going again. I am also trying to organise an in-school competition to generate some intrest. Any suggestions would be warmly welcomed.

Desmond Hamilton
Maths Department
Greenwood Academy
Dreghorn
Ayrshire

I am interested in making a video film with 480Z colour graphics. I would like to avoid actually filming the screen and would like to save the graphics directly onto tape. It is impossible to transfer graphics directly without using a PAL board which is not available for the 480Z. I would be grateful if anyone can help me to find a reasonably cheap answer to the problem. Price is important as it is for a school and there is very little money available for decent software, let alone more hardware!

Frank Miller
3 Norton Road
Garstang
Nr Preston
Lancs



DEPARTMENT OF CURRICULUM STUDIES
FACULTY OF EDUCATION

Chairman: O. E. Maravanyika, M.Ed.

P O Box MP 167
Mount Pleasant
Harare
Zimbabwe

Telegrams: UNIVERSITY
Telephone: 303211
Telex: 4-152 ZW

UNIVERSITY OF ZIMBABWE

29 November 1983
Spencer Instone
c/o Research Machines
PO Box 75
Oxford OX2 0BW

Dear Dr Instone

The Zimbabwe Computer Society is sponsoring a competition to be held in August 1984 for young people, mainly, but not exclusively, in school. The competition will involve the production of both hardware and software.

With a University colleague, Mike Robson, I have been asked to sit on the organising committee for this competition, generally because of our interest in computers in education but more specifically because we are about to launch a major curriculum project early next year in Zimbabwean schools in 'informatics' or information technology, in which micro-computers will play a most significant part. An element of the course we will introduce into schools will be the encouragement of project work, leading to the production of hardware and software of the type which will be involved in the competition.

We would be very interested to learn whether you have any experience of being involved in such a competition or in the organisation of microcomputer projects in schools. If so, could you possibly let us have:

(1) copies of any programmes, brochures, information sheets, etc. which indicate the type of classes involved and the type of projects which actually entered such a competition,

(2) the criteria used to judge such projects,

(3) any 'write up' of the competition, including details of the winning projects,

(4) any other information you think could be of help.

I realise how much of an imposition I am imposing by such a request, but I am sure that you will appreciate that microcomputers are available here at a much lower level than in the UK. Consequently, we are keen to get both the competition and the project work in schools 'just right'.

On behalf of all concerned I am grateful for any help you can offer.

Yours sincerely

Peter Towse

Peter Towse
Senior Lecturer

Ramblings from the Research User Group

I don't know how many of you read Jerry Pournelle in BYTE Magazine, but he's the main reason I buy it. His style is to ramble on about nothing in particular, but somehow he always manages to include an enourmous amount of news and informed opinion. Here's a piece in a similar vein - it just lacks the news and informed opinion.

Actually I'm supposed to be writing a report on the Research User Group meeting which was held on 12th. July at the Central Middlesex Hospital. Ian Maddison provided the local facilities, and we were certainly grateful for his air-conditioned lecture theatre, because outside it was, in a word, hot. Ian gave us a talk on the difficulties of training medical students to interpret x-rays, ultrasound scans, and the various other ways of taking pictures of our insides. Nothing to do with computers, really, except he's intending to use a Microsight TV system and 380Z to try to enhance the pictures.

Peter Kruger of Digithurst, who manufactures the Microsight system, spoke next. His hardware consists of a TV camera, connected to a slow-scan TV image digitiser which interfaces to the 380Z via the User port. He also has versions for other computers, like the Pet and Apple, and no doubt the 480Z before long. The digitiser has a spatial resolution of 256 by 256, with 8 bits of grey level data (i.e., 256 grey levels). However, there is apparently a problem in getting more than 6 of these bits into the computer as two of the eight data lines have to be used for handshaking. (There must be dozens of people cursing the 380Z's user port for lack of handshaking lines - RML have obviously learned that lesson because the 480Z's user port has four of 'em). So you're restricted to 64 grey levels. In my experience, that's plenty for most applications, in fact the noise level on most cheap camera tubes makes it pointless going to 8 bits.

The Microsight package also includes some software, and we saw demonstrated a quite nicely designed menu-driven program, which allowed images to be captured, displayed, and processed in various ways. The image processing is fairly low-level, at least compared to what the robot vision people are doing, and most of it seemed to be performed on a 'binarised' (i.e. pure black and white) version of the image. It was able, however, to correctly identify the outline of a series of blobs of branston pickle on a plate, which was standing in as a simulated culture dish.

The Microsight doesn't pretend to be a 'real-time' image analyser - for a start, it takes 5 seconds to digitise a single 256 by 256 image. However, there are many applications for which it's adequate, and as Peter pointed out, at £800 it might be attractive for feasibility studies before investing in some VERY expensive real-time hardware. If your interested you can phone Peter on 0223 - 208926.

Those of you who read my bit in Read Using 6 may remember me saying that RML might be marketing the 'Silicon Disc' for the 380Z. I discovered at the meeting that the situation has changed, and Digitimer are producing the board. John Smale of Digitimer, and Pete Furness, who designed the board, gave a quick demo (quick being the operative word!) and a short talk. Apparently it's called the Ramdisc this week, some American outfit has bagged the name 'Silicon Disc'. It's a single board which just plugs

into the RML ribbon cable bus, and it comes with a (floppy) disc of programs. One program "installs" the Ramdisc by patching the CP/M BIOS and adding the Ramdisc driver. This program is very simple to use - you just run it - but it will only work with CP/M 2.2, not 1.4. Thereafter, you access Ramdisc as drive `I:', exactly as if it were an ordinary disc, but about 200 times faster! A full review appears elsewhere in this issue. Pete Furness explained that, if you're prepared to delve into the mysteries of the page selection mechanism, you can access Ramdisc directly, a byte at a time, and so use it as a true random access memory. All in all, I liked Ramdisc a lot, although at £450 it looks a bit expensive, when you consider that for half that price you can buy a MAP 256K ram card for the Gemini and other 80-bus computers. If you're interested, Digitimer are on 07073 - 28347.

After lunch Eugene Sweeney and Graham Brown-Martin of RML spoke to us about new products. Eugene listed the many software products that are now available to run on the 480Z network; he also demonstrated the 480Z ROMpacks, and the 480Z discs. These will work in single or double density, and can read the 380Z single density discs, although this seemed very slow. Eugene explained this was because the discs are optimised for double-density operation, though why this makes single density operation slower I don't know. The 480Z discs are interfaced via the SIO4 serial interface, and I imagine this must knock the speed down, because my 480Z manual says the SIO4 only works up to 9600 baud. However, since I only have a manual for a 480Z, and not an actual machine, maybe I shouldn't attempt too much of a review! By the time you read this, double density discs (and upgrades) for the 380Z should be in volume release. Apparently you need an 80-column machine with 56K before you can upgrade, but only the disc controller needs changing, not the drives themselves, so it shouldn't be too costly.

I won't go into more details about what Eugene said; you can get the news much more reliably from the RML Newsletter or sales blurb. Interestingly, a quick canvas of those present revealed that non of them were using 480Z's. And that opens up a whole new can of worms, to do with RML's current marketing priorities and how they don't map too well onto the needs of the research user. RML have made no secret of the fact that most of their effort of late has been directed at the 480Z and the network, and in fairness to them, they probably have the best CP/M networking system you can buy. No doubt the schools will love it. However, there of those of us who don't want networks. We can remember the days of pushing batch jobs through a mainframe, or using a minicomputer too big to take out of the room, and we are delighted to now have reasonably powerful, portable machines we can take where we want and which we don't have to share with anyone else. For us, networking would be a big step backwards. Networking aside, I am yet to be convinced of the 480Z's suitability for lab work, because of the restrictions on what you can interface to it. The problem is, you can't hang anything extra onto the bus, so there's no way, for example, you could use a PIO or a CTC. True, there's a CTC inside the machine, but it's fully committed to internal functions. There's a user port, and it's better than the 380Z's user port because it's got handshake lines, but you can't generate interrupts through it. I know RML are thinking about a lab interface box to hang on the 480Z user port; I'll be interested to see what they come up with. One bright light on this particular horizon is the 480Z's optional IEEE 488 interface; it would be nice to hear from anyone who is actually using it.

Despite gentle probings, RML are reluctant to reveal their plans for a new 16-bit machine. One is tempted to wonder (perhaps uncharitably) if this is because they don't have any. Eugene did indicate that it will be a single-user machine (able to act as a network server?) and may run MS-DOS or CP/M-86. "It will not", he announced, "use existing technology". If we discount the possible interpretation that they are proposing to build a machine out of non-existant technology, my guess is they're looking at the Intel iAPX186 processor, one of the 8086's big brothers. If I'm right, I hope they don't forget the 8087 numeric co-processor, at least as an option. It seems important, however, that they should get the machine out pretty quick; the 380Z is beginning to look expensive compared with most of the other Z80 CP/M machines, and there are plenty of competitors already around to capture the 16-bit market. It will be interesting to see if they go for upward compatibility with the 380 and 480Z, or for compatibility with the IBM PC, or whether they can manage both!

Anyway, back to the meeting. Our final speaker was Brother Henry, of Mayfield college, who demonstrated a range of programs designed to diagnose and aid children with specific learning disabilities, such as dyslexia. Br. Henry gave us a good insight into the difficulties such children faced, and stressed the need to diagnose the problems early, and not merely brand such children (many of whom are actually rather bright) as "stupid". Many of his programs made good use of the high-res graphics; although they were also written with a view to compatibility with the other micros which are popular in schools, such as the BBC machine and the Spectrum. Many were written or translated by children at the college. In fact, Br. Henry's demo nearly didn't happen, because he'd brought his programs on a 5-inch disc and no-one (like me, for example) had thought to provide an MDS machine. Fortunately, RML let us try the programs on their 480Z, and lo and behold - they worked fine. I thought that was an excellent silent testimony to the degree of compatibility RML have achieved between the two machines; in fact, it was one of the more impressive aspects of the day.

The next Research User Group meeting will be held at Leicester some time in July. Notification of the meeting will be mailed to all on the Research User Group Membership list. If you aren't on it but want to be, write.

Finally, may I present my offering for the single line program competition. The point is, it doesn't work!

```
10 PRINT INT(2↑0)
```

Chris Brown,
Dept. of Psychology,
Sheffield University,
Sheffield S10 2TN.

This article would have appeared six months ago but for a blockage in the Read Using pipelines. It is included now since there is little that is out of date - it certainly shows how slowly things move in the (RML) computer world. Besides, I too am an admiring reader of Jerry Pournelle's "User's Column" and bid anyone to ramble in Read Using who wants to. -- Ed.

Software Library Reviews

Here are reviews of three more volumes of the user group software library. If you have managed to avoid learning what this library is then you could browse over the last two issues of Read Using. Or you can get details from the librarian by sending an A4 envelope stamped to the tune of 23p to:

Nick Wedd
Genetics Laboratory
Department of Biochemistry
South Parks Road
Oxford

Thanks to Peter Clarke for the review of volume three which appeared in the last issue uncredited for excessively technical reasons.

Review of Library Volume RMNUGL 1

by Nicholas Wedd.

BASSORT	COM	Basic interpreter with sort call.
COPY5	COM	Minifloppy format-and-copy utility.
CREATE	BAS	Creates index of disk directories.
DISSAMBR	ASC	Basic disassembler. Doesn't work.
GREP	COM	Unix-type search-and-display utility.
GREP	HLP	Help for above.
INDEX	R01	This file.
INPUT	COM	Opposite of TYPE: allows easy file creation.
INPUT	HLP	Help for above.
INPUT	ZSM	Source of above.
PRINTDIR	BAS	Basic program to see a directory.
REVIEW	R01	Review of things on this disk.
RND256	BAS	Some random strings, for sorting.
RNDSRT	BAS	A sort program using BASSORT.COM.
S	INF	Information about BASSORT.
SEARCH	BAS	Searches indexes created by CREATE.
SLOW	COM	Slows down programs.
SLOW	HLP	Help for above.
SLOW	ZSM	Source of above.
SRT	INF	Same as S.INF.
SRT2	BAS	Another sort using BASSORT.
TAIL	COM	Unix-type utility types tail end of file.
TAIL	HLP	Help for above.
UPDATE	BAS	Updates indexes created by CREATE.
ZDUMP	COM	Displays disk sectors in hex and ascii.
ZDUMP	HLP	Help for above.

GREP.COM is a UNIX-like utility which searches a text file for a specified string, and displays all lines in which that string occurs. Its use is explained in GREP.HLP.

TAIL.COM is a utility which displays the last portion of a file. By default, what it displays is about a screenful; but you can specify the number of records to be displayed. It requires CP/M 2 or more. Its use is explained in TAIL.HLP .

ZDUMP.COM is a utility program to list scteors of files to the screen in Hex and Ascii. Its use is explained in ZDUMP.HLP.

These were donated by John Ward.

BASSORT.COM is an RML BASIC interpreter with a fast string-sorting routine added. Full instructions for its use are in the file S.INF, which is also present in a version SRT.INF which, being in Wordstar format, looks peculiar when typed.

RND256.BAS is a collection of random strings, which can be sorted.

RNDSRT.BAS is a BASIC program that uses the BASSORT interpreter and sorts a file of strings into order. So, apparently, is SRT2.BAS.

CREATE.BAS Creates a file on one disk which contains the directory of another disk. UPDATE.BAS can be used to update this directory as files are changed on the second disk, and SEARCH.BAS to search through the directories held on the first disk for a particular file. Together, these programs costitute a very useful package for those who are inclined to forget where they put a file. The librarian certainly intends to use them.

These were donated by Colin Watkins.

SLOW.COM is a small machine-code program for slowing down, by a specified amount, any program which is subsequently run. The assembler source for it is given in SLOW.ZSM, and its use is explained in SLOW.HLP.

This was donated by Nicholas Wedd.

INPUT.COM is a simple utility for inputting directly to a file from the keyboard. Its use is explained in INPUT.HLP, and its assembler source is given in INPUT.ZSM.

PRINTDIR.BAS is a basic program for printing a disk directory.

COPY5.COM is a utility which copies a whole five-and-a-quarter-inch single-density disk side, and formats the destination disk at the same time.

DISSAMBR.ASC is a disassembler, written in BASIC. It does not work under Research Machines BASIC.

Review of Library Volume RMNUGL 2

Stephen Rochford

ACYDUCY	BAS	A primitive card game.
APOLLO	BAS	See a spaceship take off. Odd idea.
BANNER	BAS	Program to print big letters on a printer.
BIO	BAS	Prints graphs of biorhythms.
BLACKJAC	BAS	A blackjack game.
BLKFRI3	BAS	Stock market game for up to four.
BLKJK	BAS	A poor version of blackjack.
CIVILW	BAS	American Civil War game.
DIAMONDS	BAS	Prints pretty patterns.
ENTEP	BAS	Draws a picture.
FOOTBALL	BAS	American football game. Poor.
FURS	BAS	A game.
GO	COM	The empty program.
GOLF	BAS	Golf game.
GUNNER	BAS	Game like duck shoot.
HORSE	BAS	A boring game.
KING	BAS	Game: run a country for eight years.
LANDER	BAS	Another lunar landing game.
LUNAR1	BAS	Lunar landing game.
MANDALA	BAS	Draws manadalas, whatever they are.
MAZE	BAS	Draws mazes.
PINUP	PIC	A picture.
PINUP1	PIC	A picture.
POKER	BAS	A poker game.
REVIEW	R02	A review of this volume.
RUSSIAN	BAS	A particularly stupid game.
SNOOPY	BAS	Draws a picture of Snoopy.
SNOOPY	PIC	A picture.
TACOS	BAS	A program better not written in basic.
TAXMAN	BAS	A numerical game.
TRADE	BAS	A playable game.
TRAP	BAS	A numerical game.
TWEETY	PIC	A picture.
WARI	BAS	Tha African game of Wari. Recommended.
WSSEARCH	BAS	Makes word squares slowly.
WUMP	BAS	A primitive sort of adventure.

This is a brief review of all of the programs which are contained on this disc. All the programs were tested on a 380Z with COS4.0 and HRG; the HRG board was being used for extra memory (47k CPM) although most of the programs will run in a 31k system. Most of the programs were designed to use an 80 column output device of some sort so they will work well in 80 column mode and I have indicated where the output will look messy if you have only got COS3.4. It is, of course, possible to change all the PRINT statements to take into acccount the narrower screen and this could be a project for reasonably intelligent school pupils. The final problem is that many of the games askyou to type in 1 for yes and 0 for no. Again this could be changed without too much difficulty.

BLACKJACK

This is the standard card game; it is all text and is messy in 40 characters. It has 0/1 for no yes input but it does work and it could be improved as mentioned above.

CIVILW

This is a simulation of the American Civil War. I must admit to very little knowledge of the actual course of the war so I cannot say if tthe game is an accurate simulation. It does work although again it looks better in 80 columns but otherwise it seems quite good. If it is an accurate portrayal of the war then it might be useful for history lessons.

FOOTBALL

Once again a subject of which I have no knowledge! Even if you do know American Football you would probably need to write down the instructions as each type of play has a number code; 11 for pass, 12 for sweep etc. The game could be used to demonstrate the use of standard functions; almost all of them are used including a "real time clock" generated using RND! Again the game could be improved for 40 column screens.

GOLF

I just about understand golf and I do not think this is a good simulation. Of course its major problem is that it is using words to describe what is really a game that needs pictures. It also seems to keep taking clubs out of the bag that ought to be there! The only consolation is that once you get to the green you seem always to get the ball in, no matter how hard or soft you hit it.

GUNNER

This makes use of basic physics; a gun aimed at an elevation of 45 degrees gives maximum range with the range decreasing for angles on either side of that. A random number is picked as the distance to your target and you are asked to pick an elevation for the gun. Unfortunately if you do not get the angle exactly right (and this means working in decimals!!) you will never win. The game allows you an error margin of 100 yards; increasing this (change line 460 to IF ABS(E) <500 THEN 310) gives you a better chance.

LANDER

Lander simulates a moon landing and asks you to give values for the thrust of your engines, the attitude of your craft and the time for which you want this to be maintained. It is not an easy game and it is accurate; no RND used anywhere to introduce elements of guesswork! It is virtually impossible to lay on a 40 column screen but it will let you work in metres (sorry, metERs !) or feet and yes/no answers are taken as Y/N.

POKER

This another text only card game; it does work but it really needs an 80 column screen. The yes/no questions do accept y/n.

SNOOPY.PIC, TWEETY.PIC, PINUP.PIC, PINUP1.PIC

Theses are all pictures, the contents of which should be obvious. You can look at them on an eighty column screen but you really need to print them out. If you only want 1 copy then make sure that the printer is set up and then from CPM (i.e. when you have the A> prompt) do the following: press ^P (this means press the CTRL button and P at the same time) and then type: TYPE SNOOPY.PIC and then when the picture finishes type ^P again.

FURS

This is a game in which you are the leader of a fur trading expedition: you choose the number of each type of fur that you will carry and where you want to sell them. Again it is a game that works without being very exciting.

HORSE

This is a simulation of a horse race; the horses are always the same and unless you are a desperate gambler the game is not very exciting. In spite of that it does work and it could be improved - another school project?

KING

This is the kingdom game where you are asked to rule your country for 8 years, making decisions on how much of your land should be industrial and farm; how much you should pay your workforce etc. It could be used seriously but I doubt that it is really accurate enough and although it works it is rather boring to play as a game.

LANDER

This is a lunar lander game. If you have COS 4.0 and you have connected up a speaker then several lines will cause bleeps; if youahve not got COS 4.0 or you haven't connected a speaker then these lines will simply cause the cursor to flicker; don't worry! As ever you are asked for a burn rate and time and your height, speed and remaining fuel are displayed. If you wanted to you could use this as a serious Physics experiment; or you could try and add some graphics to it (a rather more advanced school project but not impossible!)

MANDALA

This prints mandalas. You can meditate on them while you are wondering what all the questions mean!! Seriously though they do produce some occasionally pretty pictures.

MAZE

This prints a maze; print 30 of them and keep a class happy for ages! You need to have a printer for this so it doesn't matter if you only have a 40 column screen

RUSSIAN

Russian roulette with a computer????? Well at least you don't get blood on the walls and the pictures are quite good!

SNOOPY This prints a picture of Snoopy!

WUMPUS

You are asked to hunt for the "wumpus", a creature which lives in a 20 room cave with three passages leading from each room. At any time you can try shooting a crooked arrow or simply move. You are also threatened by bats, who will move you at random, and by bottomless pits - if you fall into one of these you die. The game is a bit difficult if you find it hard to visualise a 3-D cave,but the game does work and you can use it on a 40 column screen.

ACYDUCY

This is a game in which the computer deals two cards and you are asked to bet that the next card will be in between the two cards already dealt. It's not the world's most stimulating game (particularly if the cards are 4 and 5 !!!) but it does work and it can be used with a 40 column screen.

APOLLO

This program starts off with a countdown and then Apollo takes off; the picture of takeoff is quite good. Then ... well try it and see! As it sands it will not really work on a 40 column screen but it could be changed fairly easily.

BANNER

This lets you print banners (which can be much bigger than you could fit on an 80 column printer!). You are offered 2 methods for printing the first takes the letters of your words and uses them to make the big letters (e.g. an F would be made up from many F's) or you can type in any two characters and these will then be used for everything. The only big problem with the program is that it copes poorly with diagonals.

BIO

This plots a graph of your mental, emotional and physical biorhythms. If you believe in biorhythms then this program is good; even if you don't it's the sort of program that could be used, with a printer, at a school fair or something like that. It is not really possible to convert it for a 40 column screen.

BLKFRI3

This is a stock market simulation in which you are asked to invest a total of $5000 in 10 different securities over a maximum of ten years. Four players can play the game at the same time; it looks quite a good idea for a painless way of playing the stockmarket.

BLKJK

This is yet another game of blackjack and although it does accept words as input, rather than code numbers and it will run on a 40 column screen, once you are past the instructions, it is not very good in terms of layout but at least it works.

DIAMONDS

This prints pretty patterns; again it really needs a printer so that you get something to keep. You can change the RML for any letters that you want and it will still work quite happily!

ENTEP

This prints a picture of the USS Enterprise; again a printer is needed; because the program uses LPRINT all the way through if you actually want to display it on the screen then type PRINTER 0 before you start.

TACOS

This is one of the pointless programs which serve no purpose except giving people who prefer PASCAL etc a further excuse to denigrate BASIC!

TAXMAN

This has no real relation to taxmen except that the computer always wins. It actually tests your ability to factorise numbers and as such is quite good.

TRADE

This is a game of "interstellar trading". It will not run in 31k but if you have got HRG then you can use a 47k CP/M and this will let you play the game. I have not played the whole thing but it looks reasonably good.

TRAP

This is a number guessing game; the computer thinks of a number and you are asked to think of a pair of numbers which will trap its number. For example 45 and 60 would trap the number 55. It does work and it does get some children thinking logically about numbers.

WARI

This is the classic game of awari. I say that because if you don't know it then you won't play it! RML computers used to come with a version of this which included instructions. If you have the instructions then I'm sure it might be good!

WSSEARCH

This makes wordsquares. If you don't mind waiting what seems like a long time then the program does work although you obviously need a printer otherwise it isn't much fun. You can use wordsquares as a way of introducing new words to a class and if you like this sort of thing then this program is alright.

Review of the contents of library volume RMNUGL 4

by N. Wedd.

ANTONYMS	BAS	English-teaching game.
BEARINGS	BAS	
CHAR	BAS	Displays HRG character set.
DEFCHAR	BAS	Sets up a matrix for defining a character.
DOTS	BAS	Plays dots-and-boxes - playable.
DRAGRACE	BAS	Poor game.
FOOTBALL	BAS	American football. Poor.
GOLF	BAS	Barely playable.
GREEKRTS	BAS	Learn Greek roots through this game.
HAMURABI	BAS	You run a country.
HANGMAN1	BAS	Hangman game - guess the letters in a word.
HANGMAN2	BAS	Ditto.
HANGMAN3	BAS	Ditto.
HIDESEEK	BAS	Good mathematical game.
HIRES1	BAS	Geography learning with HRG.
HIRES2	BAS	Draws cones with HRG.
HIRES3	BAS	HRG screen effect.
IQUEEN	BAS	Very poor game.
LOWRES1	BAS	Boring low-resolution-graphics picture.
MAN	BAS	HRG man "runs" across screen.
MAZE	BAS	Draws good mazes slowly.
MEDRES1	BAS	HRG screen effects.
MEDRES2	BAS	HRG screen effects.
MEMBRAIN	BAS	Cell biology game.
NOEL	PIC	Picture of Virgin and child.
POSTERS	BAS	Big sideways writing on a printer.
PRDUMP	BAS	Microline 80 screen dump.
PROG1	BAS	Draws a cardioid in HRG.
REVERSE	BAS	A well-written puzzle.
REVIEW	R04	Review of this volume.
ROULETTE	BAS	Roulette. You lose every bet.
SHOP	BAS	Strongly recommended.
SNOOPY	BAS	Draws a picture of Snoopy.
SNOOPY	PIC	A picture of Snoopy.
SPACIN	COM	Fast space invaders.
STARTREK	BAS	There are those who enjoy this.
TABCAR	BAS	Simple tables-learning game.

ANTONYMS.BAS is an English-teaching game in which you have to guess the opposite of the word offerred by the machine. There are often several acceptable answers. Motivation is provided by the fact that you cannot move ot to the next question until you have answered the previous one correctly - I got stuck on the opposite of "anathema", and cheated by listing the program.

DOTS.BAS is the old game of dots-and-boxes, played against the machine. A playable game, but quite easy to beat.

DRAGRACE.BAS is a game in which you "design" a car by answering incomprehensible questions about tyre dimensions, etc., and then race it against the computer's car, getting a message such as "car 2 stops burning rubber". I doubt this can be worth playing even if you know something about drag-racing.

FOOTBALL.BAS Another game of the type-meaningless-numbers-until-you-lose variety. This one has things called "intercept" and "screen pass", so I suppose it is American football.

GOLF.BAS Similar to the above, but the number you choose controls the club you are to use, so at least you know what is meant to be going on. No pictures or anything like that. A peculiarity of this version is that every time you hit the ball, it says "Syntax error at line 1690". This appears to be cured just by deleting line 1690.

GREEKRTS.BAS An educational game which lists some Greek word roots, and then asks you which one has a certain meaning. The original Greek list soon scrolls off the top of the screen, so its helps to remember them. It is fussy about giving being given the right amount of a root: I gave it geos when it wanted ge- and anthro- when it wanted anthropos, both were wrong.

HANGMAN1.BAS is the well-known game in which you have to guess the letters in a word of which you initially know only the length. A wrong guess and you lose a life, a right guess and you get to see all instances of that letter in the word. Lose 10 (I think) lives and you see yourself hanged.

HANGMAN2.BAS and

HANGMAN3.BAS are similar, but with different vocabularies.

HIDESEEK.BAS is a game in which, in effect, four things are hidden at the centres of squares on a ten-by-ten board. You name a square, and are given the distances, to two decimal places, to each of the four things. You have ten guesses. When you guess the square that a thing is on, you have found it; and you try to find all four things within your ten guesses. If you can apply Pythagoras, and are awake, it should only take five or six guesses; but in practice, one tends to get confused. I would say that this would help to teach a bright pupil about co-ordinate geometry (and Pythagoras) while being just challenging enough to maintain motivation.

IQUEEN.BAS It is of some interest to sit down and analyse this game, but none to play it. It makes things worse that in this version, you cannot see the board that you have to move on.

MAZE.BAS draws a maze of the size that you specify. It is slow for large mazes, but not as slow as other versions I have seen. The mazes it draws are sensibly designed: it seems to have the same ideas as I do about the connectedness, etc., that a maze should have. Its logic should be worth studying.

MEMBRAIN.BAS is a game in which you have to keep a cell alive by moving molecules in or out. The instructions are poor: at first, when I wanted more of a substance in the cell, it would only offer to remove it. When you get the hang of it, you can stay alive for a bit, but it seems that you always die eventually. So it is rather uninspiring.

ROULETTE.BAS asks you how mmuch money you want to bet, whether on a number a colour, etc., and which number, colour, etc. I could not tell whether it had the right odds for everything, as I lost every bet I made. I presume it was written by a very simple-minded puritan.

SHOP.BAS	This is not exactly a game. You are visiting a food shop, and make various purchases. It is good at telling you what options are available. It can also be very funny, particularly in the way it gives sensible and amusing responses to input which would make poorly-written programs give up. This seems to be done, not by clever programming, but by a programmer who has anticipated the sort of input it is likely to get. Definitely recommended, very enjoyable to play.
SNOOPY.BAS	draws a picture of Snoopy.
STARTREK.BAS	Another of the type-meaningless-numbers-until-you-lose variety. I actually managed to shoot down one Klingon, completely at random. But maybe computer startreks have enough adherents that there are people who will understand this one.
CHAR.BAS	draws the character set in HRG. Requires HRG level 2.
DEFCHAR.BAS	draws a 10 by 8 array on the screen, and lets you design a character in it, telling you the values of the bytes needed to define this character in COS 4. No underline is offerred.
HAMURABI.BAS	You are the ruler of a small underfed country and must decide whether to buy or sell land, how much grain to plant, etc. Not much in the way of instructions, but that is deliberate here: you learn from experience. And even if you kill off half of your population in the first two years, you can still manage a recovery. Playable.
HIRES1.BAS	Requires HRG level 2. It draws a map of England and Wales, puts six towns on it, and lists their names. You then have to guess, or remember, which town goes with which name. This seems to me to be a good way of learning: maybe the first time Brighton appears, all you know is that it is in the South, and you can work out which it is by elimination. Then when Brighton and Southampton both appear in a later frame, you remember where you saw Brighton before.
HIRES2.BAS	asks you for some numbers - it is not at first clear what it intends to do with them - and then draws something like a cone. Requires HRG.
HIRES3.BAS	Pretty screen effect. Requires HRG.
LOWRES1.BAS	uses low-resolution graphics to draw an uninteresting picture.
MAN.BAS	draws a small picture of a man, who then proceeds jerkily across the bottom of the screen. Requires HRG.
MEDRES1.BAS	Screen effects. Requires HRG.
MEDRES2.BAS	Ditto. Ditto.
NOEL.PIC	A picture of a virgin and child.
POSTERS.BAS	Creates large sideways writing on a printer. You specify the size and the message. Useful for demos?

PRDUMP.BAS does a screen dump onto a microline 80 printer. It works on an Epson but fills out lines with plus signs.

PROG1.BAS draws a cardioid. Requires HRG.

REVERSE.BAS a simple but well-written and pleasing game, which requires you to put a set of jumbled digits into order. Uses the screen well.

SNOOPY.PIC A picture of Snoopy.

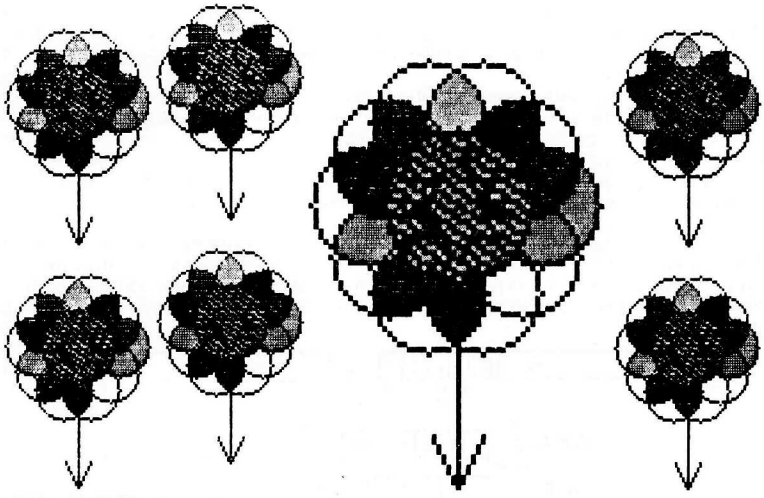
SNOOPY2.PIC Another picture of Snoopy. Do people have nothing better to do with their time?

TABCAR.BAS is a multiplication-tables testing game. It asks you a tables question: get the answer right and your car (drawn in low resolution graphics) moves forwards, wrong and the computer's moves forwards. Some of the questions are very easy: what is 3 * 1 ?

SPACIN.COM Not in BASIC, this one, but in fast machine code. It is a game of Space Invaders, looking as much like the arcade original as low resolution graphics and a forty-column screen allow. It is very fast: without cheating, the reviewer could never survive beyond two seconds.

It differs from the original in several ways. You only get four lives all together instead of three per frame. You do not have to clear a frame completely to move on to the next one. The invaders do not move systematically and relentlessly across the screen, but mill about aimlessly, tending to avoid your laser. There is no mother ship moving rapidly across the top for bonus points.

N.B. This works with COS 3.4, but NOT with COS 4. I do not know why.



Memory space is saved by using a single byte code for each Basic command word when a program is stored in memory. Basic version 5.0 commands are given the codes from 128 (denary) upwards towards 255 and these are displayed in illustration 1 with the program required to find them out. The hex address of the position in memory where the code words are stored in a 'lookup' table is also given for each word.

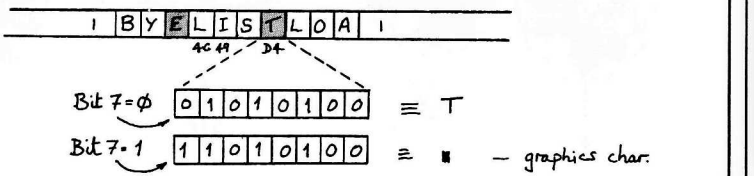
Because of the number of command words now in our more sophisticated version 5.0 Basics there are not enough numbers to allow the SBAS21 codes to be put in the same table so SBAS codes have to be given numbers already allocated. Two bytes therefore are required to show an SBAS code to avoid confusion. So for example 255 followed by 139 stands for the SBAS command 'ENDLOOP' (FF 8B in hex), whereas 139 on its own stands for 'GOSUB' (8B in hex). All the SBAS21 codes, including Random Access Files, are shown in Illustration 2, along with the program needed to find them out. Note, you require SBAS21 to be loaded before this program will work.

Both these sets of command words are stored as collection of ASCII characters in lookup tables within the Basic interpreter memory area and to save room the last character of each word is coded with an extra 128 added (so bit 7 is '1') to indicate the end of the word. See illustration 3, this shows that the command word 'LIST' is stored as 4C 49 53 D4. When a 'T' is not the last character in a word it is stored as bit pattern 01010100, whereas when it is the last character it is stored as 11010100, which is a white graphics character as shown. List is stored at hex address 2576, see illustration 1.

If you search for an ASCII code pattern for LIST ,using the G command in the front panel mode, (G 4C 49 53 54 followed by N commands to find the Next occurance of the pattern) none is found in the program area. But if you search for 'LIS' a patern match is found and the next byte is seen to be 'T+128' a graphics character. When working at this fundamental level a list of the ASCII codes in hex and denary is most useful, see illustration 4 for the ASCII codes and a program to generate them.

A line of Basic program is stored in memory in 'internal format' not as ASCII characters as seen on the screen. All command words are stored as single byte codes as mentioned and the rest as ASCII characters, notice however that the mathematical opperators + - * / and others have command codes and not ASCII codes, this is important later!

At the front of each program line in memory are two bytes, which point to the address of the next line, followed by two bytes which are the line number in four digit hex notation. A single zero byte follows at the end of the line as a delimiter.



```
5 COUNT=128
10 ST=&246E : ?HEX$(ST);"  ";
20 X=PEEK(ST)
30 IF X)127 THEN GOTO 200
40 PRINT CHR$(X);
50 ST=ST+1
60 GOTO 20
70 :
200 X=X-128
210 PRINT CHR$(X);"  ";TAB(17);COUNT
220 ST=ST+1 : COUNT=COUNT+1
225 IF COUNT=256 THEN STOP
230 ?HEX$(ST);"  ";
240 GOTO 20
999 END

Ready:
```

Illustration 1.

```
RUN
246E  END          128   252E  RESUME     172   25D1  SQR        216
2471  FOR          129   2534  OPEN       173   25D4  RND        217
2474  NEXT         130   2538  CREATE     174   25D7  LOG        218
2478  DATA         131   253E  CLOSE      175   25DA  EXP        219
247C  INPUT        132   2543  FILES      176   25DD  COS        220
2481  DIM          133   2548  RESET      177   25E0  SIN        221
2484  READ         134   254D  ERASE      178   25E3  TAN        222
2488  LET          135   2552  RENAME     179   25E6  ATN        223
248B  GO TO        136   2558  DIR        180   25E9  PEEK       224
2490  IF           137   255B  BYE        181   25ED  LEN        225
2492  RESTORE      138   255E  RUN        182   25F0  HEX$       226
2499  GO SUB       139   2561  LOAD       183   25F4  STR$       227
249F  RETURN       140   2565  MERGE      184   25F8  VAL        228
24A5  REM          141   256A  NEW        185   25FB  ASC        229
24A8  STOP         142   256D  SAVE       186   25FE  CHR$       230
24AC  PUT          143   2571  FSAVE      187   2602  LEFT$      231
24AF  OUT          144   2576  LIST       188   2607  RIGHT$     232
24B2  ON           145   257A  LLIST      189   260D  MID$       233
24B4  WAIT         146   257F  RENUMBER   190   2611  POINT      234
24B8  DEF          147   2587  DELETE     191   2616  ERR        235
24BB  POKE         148   258D  EDIT       192   2619  ERL        236
24BF  PRINTER      149   2591  CONT       193   261C  VARADR     237
24C6  PRINT        150   2595  TO         194   2622  GET        238
24CB  ?            151   2597  FN         195   2625  ATTRIB     239
24CC  PLOT         152   2599  THEN       196   262B  LOOKUP     240
24D0  GRAPH        153   259D  NOT        197   2631  BREAK      241
24D5  TEXT         154   25A0  STEP       198   2636  GO         242
24D9  CLEAR        155   25A4  TAB (      199   2638  )          243
24DE  ERROR        156   25A9  SPC (      200   263A  c          244
24E3  ELSE         157   25AE  +          201   263B  U          245
24E7  LPRINT       158   25AF  ^          202   263C  Y          246
24ED  TRACE        159   25B0  *          203   263D  c          247
24F2  LTRACE       160   25B1  /          204   263E  Y          248
24F8  RANDOMIZE    161   25B2  -          205   263F  ]          249
2501  LLVAR        162   25B3  AND        206   2640  e          250
2506  LVAR         163   25B6  OR         207   2641  e          251
250A  CALL         164   25B8  SGN        208   2642  U          252
250E  LINE         165   25BB  INT        209   2643  E          253
2512  EOF          166   25BE  ABS        210   2644  e          254
2515  QUOTE        167   25C1  USR        211   2645  ).         255
251A  WIDTH        168   25C4  FRE        212
251F  LWIDTH       169   25C7  INP        213
2525  NULL         170   25CA  LPOS       214
2529  LNULL        171   25CE  POS        215
```

```
  3 PRINT"COMMAND WORDS AND CODES FOR SBAS21":?:?:
  5 count=128
 10 st=&37ae : ?HEX$(st);"  ";
 20 x=PEEK(st)
 30 IF x)127 THEN GOTO 200
 40 PRINT CHR$(x);
 50 st=st+1
 60 GOTO 20
 70 :
200 x=x-128
205 h$=RIGHT$(HEX$(count),2)
210 PRINT CHR$(x);"  ";TAB(21);count;"  ";h$
220 st=st+1 : count=count+1
225 IF count=199 THEN STOP
230 ?HEX$(st);"  ";
240 GOTO 20
999 END

Ready:


RUN
COMMAND WORDS AND CODES FOR SBAS21
```

Illustration 2.

37AE	FORMAT$ (	128	80	3864	DISPLAY	164	A4
37B7	FOR	129	81	386B	LNAMES	165	A5
37BA	NEXT	130	82	3871	NAMES	166	A6
37BE		131	83	3876	WARN	167	A7
37BF	REPEAT	132	84	387A	DELETEPROMPT	168	A8
37C5	UNTIL	133	85	3886	SWAP	169	A9
37CA	WHILE	134	86	388A	CURSOR	170	AA
37CF	ENDWHILE	135	87	3890	FETCH	171	AB
37D7		136	88	3895	PRESERVE	172	AC
37D8	DO	137	89	389D	OLDLOAD	173	AD
37DA	LOOP	138	8A	38A4	LOADGO	174	AE
37DE	ENDLOOP	139	8B	38AA	MERGEGO	175	AF
37E5		140	8C	38B1	LEVEL	176	B0
37E6		141	8D	38B6	TIMES	177	B1
37E7	ENDCASE	142	8E	38BB	GARBAGE	178	B2
37EE	CASE	143	8F	38C2	DEGREES	179	B3
37F2	WHEN	144	90	38C9	BASE	180	B4
37F6	OF	145	91	38CD	LOCAL DATA	181	B5
37F8	OTHERWISE	146	92	38D7	KEEP	182	B6
3801	PROCEDURE	147	93	38DB		183	B7
380A	PROC	148	94	38DC		184	B8
380E	ENDPROC	149	95	38DD		185	B9
3815	FUNCTION	150	96	38DE	RCREATE	186	BA
381D	ENDFUN	151	97	38E5	RINIT	187	BB
3823	RESULTIS	152	98	38EA	ROPEN	188	BC
382B	GLOBAL	153	99	38EF	RREAD	189	BD
3831	RECOGNIZE	154	9A	38F4	RWRITE	190	BE
383A	KILL	155	9B	38FA	RCLOSE	191	BF
383E	ENDIF	156	9C	3900	RFILES	192	C0
3843	BEGIN	157	9D	3906	TRUE	193	C1
3848	END	158	9E	390A	FALSE	194	C2
384B	EXIT	159	9F	390F	BOUNDS (	195	C3
384F	ELIF	160	A0	3917	PAD$ (	196	C4
3853	CAPS	161	A1	391D	UNPAD$ (	197	C5
3857	BECOMES	162	A2	3925	\	198	C6
385E	INDENT	163	A3				

```
 10 ?"Print out of the ASCII codes and characters"
 20 ?"*****************************************"
 25 ?:?
 30 '
 40 st=32 : count=1
 50 ch$=CHR$(st)
 60 h$=HEX$(st)
 70 h$=RIGHT$(h$,2)
 75 IF count=2 THEN GOTO 200
 80 ?ch$;TAB(5);st;TAB(10);h$;
 90 st=st+1 : count=count+1
100 IF st=129 THEN STOP
110 GOTO 50
200 ?TAB(30);ch$;TAB(35);st;TAB(40);h$
210 st=st+1 : count=1
220 IF st=129 THEN STOP
230 GOTO 50
999 END

Ready:
run
Print out of the ASCII codes and characters
*****************************************
```

	32	20	!	33	21	h	104	68	i	105	69
"	34	22	#	35	23	j	106	6A	k	107	6B
$	36	24	%	37	25	l	108	6C	m	109	6D
&	38	26	'	39	27	n	110	6E	o	111	6F
(	40	28	)	41	29	p	112	70	q	113	71
*	42	2A	+	43	2B	r	114	72	s	115	73
,	44	2C	-	45	2D	t	116	74	u	117	75
.	46	2E	/	47	2F	v	118	76	w	119	77
0	48	30	1	49	31	x	120	78	y	121	79
2	50	32	3	51	33	z	122	7A	{	123	7B
4	52	34	5	53	35	|	124	7C	}	125	7D
6	54	36	7	55	37	~	126	7E		127	7F
8	56	38	9	57	39		128	80			
:	58	3A	;	59	3B						
<	60	3C	=	61	3D						
>	62	3E	?	63	3F						
@	64	40	A	65	41						
B	66	42	C	67	43						
D	68	44	E	69	45						
F	70	46	G	71	47						
H	72	48	I	73	49						
J	74	4A	K	75	4B						
L	76	4C	M	77	4D						
N	78	4E	O	79	4F						
P	80	50	Q	81	51						
R	82	52	S	83	53						
T	84	54	U	85	55						
V	86	56	W	87	57						
X	88	58	Y	89	59						
Z	90	5A	[	91	5B						
\	92	5C	]	93	5D						
^	94	5E	_	95	5F						
`	96	60	a	97	61						
b	98	62	c	99	63						
d	100	64	e	101	65						
f	102	66	g	103	67						

```
Interrupted at line 100
Ready:
```

Illustration 3

Dr Philip H Hepburn

Research Machines National User Group

11 Westbourne Park Road
Westbourne
Bournemouth
BH4 8HG

REPORT of INCOME and EXPENDITURE

This report relates to all income and expenditure during the period 2nd March 1983 to the 1st March 1984.

Income	~~+++++.++~~
Balance brought forward 1st March 1983	1502.00
Subscriptions	8101.00
Courses	5647.00
Miscellaneous	66.62
Total	15316.62

Expenditure	
Cash in Hand	3.00
Courses	5261.06
Office Expenses	2394.02
Meetings	1225.96
Read Usings	242.50
Income Tax	49.20
Miscellaneous	64.42
Total	9240.16
Balance on Current Account	6076.46
Balance on Deposit Account	1684.00

S.G.Boswell

18 March 1984

Indices (?) to old Rude Easings

Due to pressure from our readers, we reproduce here the contents of previous editions of Read Using. Where an article has more than one author we have given only the first named (sorry, second and third authors!). The last column tries to give you some idea of what the article is really about, despite its title. Back copies of Read Usings are available to members at a cost of #2.50 from the Treasurer. If you want to sell any backcopies to the Treasurer, he will buy them off you for #1.50 for the first copy you send him, and #1.25 for subsequent copies in the same envelope. The exorbitant price difference between buying and selling price pays for postage and packing.

Read Using 1

Page	Author	Title	What its about
2	Angela Hirst	Mini-Z	Front Panel, a simpler one
4	J Townsend	Keyswap	Debugging tool for BASIC
8	Nick Cutler	IBM 3740	IBM 3740 to CP/M conversion
10	Morfydd Edwards	MICE	Micros in Computer Education
11	Brian Drury	Make CP/M work for you	Turnkeying disks
12		Restoring files	Restoring erased files

Read Using 2

3	Bill Tagg	Chairman's report	Chairman's report
5	Derek Esterson	Secretary's report	Secretary's report!
7	N W Richards	XDB Privacy codes	Privacy codes - how to overcome them
9		MID$()=	MID$= - putting it into BASIC 5
10	Helen Townley	Computers and Disabled	Disabled people - using computers
11		Micro Standards	Hertfordshire's standards
12	James Watson	Utility Software	Space Invaders!
15		Joysticks	Joystick interface
17	J H Carpenter	Restoring files	Restoring erased files on 8" disks
19	David Mitchell	Four voice polyphony	Polyphonic music form the 380Z
20	David Coates	PIO Plagiarised	PIO - interconnections
21	George Dawson	Labelling disks	Labelling disks by filenames
23		Crash proofing disks	XDB - handling all errors properly
25		Sticks	NIM - a program that 'learns'
29		Deviance	Social science program (light-hearted!)
31	Pat Crabb	The Nascom and 380Z	Nascoms - using them as terminals
34	Morfydd Edwards	MICE	Micros in Computer Education news
36		Disk STAT from BASIC	STAT from XDB

Read Using 3

2	Nick Cutler	Guidelines	Guidelines for Programmers
10	P Smith	SBAS	Review of SBAS - Structured BASIC
16	S P A	BASMOD	Subroutines - naming them in BASIC
17	Max Westby	Fast full disk copy	Disk-copying: for 8" disks
24	DGS/CEM	Resetting the PIO	PIO - curing a resetting bug
27	Judith Gordon	Helping the handicapped	Handicapped students at an F E College
28	J M Ridge	Fast disk copying	Disk-copying: for 5" disks
33	ASLIB	ASLIB	British Library project
36	A Turner-Smith	EMT calls from FORTRAN	FORTRAN - adding CALL EMT()
39	Peter Jowett	CP/M User Library	CP/M Library - brief review
42	P A Watmough	Batships	Battleships - a BASIC program
47	Derek Wickham	BASIC as an editor	BASIC - using it as a text editor
51	Peter Jowett	Secure Privacy	XDB - restoring the private parts!
52	J M Ridge	Extending BASIC	BASIC - adding machine code routines
63	J M Ridge	STAT from BASIC	STAT from BASIC version 5
67	P A Watmough	Findlady	Find the lady - BASIC program

Read Using 4

8	Andy Webb	A O B	Executive committee alternative minutes
10	Malcolm Neave	Croydon LOGO	LOGO - the Croydon Turtle Graphics
15	P Smith	Computing using BASIC	Review of Tonia Cope's book
16	S P A	Sound for the 480Z	480Z sound without escape sequences
18	J M Ridge	Simple Input	Text typed straight into a file from CP/M
22	Nick Cutler	Turnkey Systems	Turnkeying disks
28	S P A	Zero length files	GOing back into your program
29	Nick Wedd	Turnkey systems	Turnkeying disks
33	Nick Cutler	Introducing Captain Z	Controlling a camera
38	David Williams	Polynomial	Equation solution with Newton Raphson
41	S P A	Structured BASIC	SBAS - example programs
43	H H Jones	Extending BASIC	Dumping the screen to a printer
46	Michael Wilson	CP/M	STAT - how it works
48	B E Weller	Scientific Computing	Polytechnic of the South Bank courses
	Wilfrid Sollom	Fortran Supplement	FORTRAN - various useful additions
49	J H Ridge	Character Building	Characters - defining your own
54	P Jowett	Disk Crash Recovery	Restoring badly erased files!
61	S P A	SBAS2 Quicksort	Sorting in SBAS with a quicksort
62	Nick Cutler	Shell Sort	Sorting in BASIC with a shell-sort
64	P G Lewis	PIC	Graphics - facilitating their use
67	Nick Cutler	Multiple hard copies	Copying and printing methods
70	Harry Morten	Incorporating STAT	STAT from BASIC - 8" systems
72	P J Butt	LIST.ALG	ALGOL - listing programs from it
74	Harry Morten	Super directory	Ambiguous file name handling from BASIC

Read Using 5

7	Nick Wedd	Understanding Assembler	Assembly language - an introduction
12		Input routine	Inputting from CP/M without EMT's
17	Jim Reddaway	Typing to disk	Inputting to a file using PIP
18	John Ward	BCPL	Review of Richards Computers BCPL
24		DATE.B	DATE.SYS file - updating it
26	Max Westby	Diabolical Hard Copy	Dumping HRG and LRG on a Diabolo
36	James Watson	Using the CTC	Counter-timer chip - using it
38	Nick Wedd	380Z Sound production	Sound from an 80 character 380Z (Correction in RU6)
41	P G Lewis	Computer dating service	Dating a disk
42	Nick Cutler	Compiling MBASIC	Microsoft's compiling BASIC
50	J M Ridge	Images of Memory	HRG - speeding it up using cache
52	W M Aitken	PRINTDIR.ZSM	STAT both sides and print them
57	Nick Wedd	Speeding up CALLs	Adding CALLs to BASIC (quick version)
58	Don Smith	Copy screen to printer	Dumping the screen to an Epson MX80
64	Robert Wills	Sorting large Databases	Sorting multiple files with CBASIC
69	P G Lewis	Superdirectory	XDB directory file
70	Bill Aitken	Epson Utilities	STAT both sides of a disk
71	Bill Aitken	Epson Utilities	Listing long files in 2 columns
76	P A Watmough	BEARINGS	Bearings with a BASIC program
81	Nick Wedd	Linkup	Review of 380Z terminal software
82	Steve Thomas	From Bach to Byte	Music board for a 380Z
88	D G Whitehouse	School Administration	Administration program for DES and ISIS
94	J H Ridge	Pretty printing	Indenting BASIC listings
100	H Morten	Yet more on Turnkey	Turnkeying disks again (FDS)
105	James Watson	Text Format checker	80 line HRG text
106	Bill Aitken	Patching CP/M	DIR - adding free disk space info.
108	Jonathan Darby	Is the 380Z a good buy?	380Z/480Z comparison
110	D J Smart	AUTOBOOT.ZSM	Turnkeying disks
112	W E C	Not a review	Micros in Schools awareness pack
114	Solomon Tomides	An alternative review	Review of Tonia Cope's book
116	Mike Blandford	Double your disks	Single siding your disks
118	Nick Cutler	Laboratory Micro	Review of Laboratory Microcomputer

Read Using 6

7	N R Wardell	Patches for XDB	XDB - skip Highest memory, disable CTRL-T
9	John Richards	BCPL reply	BCPL - reply to review
11	Richard Syrett	RML User Group Library	Library facilities for the User Group
13	D R Ellis	Why not CALL "DRAW"?	Drawing CALL from BASIC
17	Nick Wedd	TXED: preparing a macro	TXED macros prepared on the screen
18	D Peacock	A language comparison	Languages on the 380Z - a comparison
36	Nick Wedd	Swapping colours	Colour improvements by swapping leads
37	Peter Butt	LOGO - first lessons	Review of RML LOGO
39	Chris Brown	Research User Group	Research User Group report
41	Richard Naylor	First chain experiences	Review of the Network
43	Jonathan Darby	Review of the Network	Review of the Network!
45	Robert Wills	Using a link	480Z as a printer buffer
53	Jon Sheppard	Copying discs	Copying disks from BASIC
54	Anon	OBOC	TXED quirks
55	Philip Hepburn	Letter from Gwent	Indexes from BASIC
61	Philip Hepburn	Wordstar	Wordstar - Practical advice
64	A D Petford	Hardcopy graphics	Dumping HRG on an Epson
69	Nick Wedd	Initialising a printer	Printer initialisation from CP/M
70	J C Ward	Make your 380Z noisy	Sound from a 380Z
74	Nick Cutler	Output to a scope	Oscilloscope receiving 380Z output
75	W J Shore	Adding things to BASIC	Adding machine code CALLS to BASIC
78	Nick Wedd	Use of odd corners	CP/M - finding a bit of room
79	Briget Harrison	Computing using BASIC	Review defence of Tonia Cope
80	J G Hunniford	PRINTSET	Printing Microquery output nicely
81	Nick Wedd	Moving the stack	Stack - moving it out of the way
82	A P Bray	Epson LRG dump	Dumping LRG on the Epson
87	Bill Aitken	MCOPY.ZSM	Copying a program onto several disks
92	H H Jones	Enhancing DIR display	DIR - putting output into columns
97	Nick Wedd	Correction	Sound article RU5 - correction

Read Using 7

2		One line programs	Competition results - one-liners
4	G Dale	Extended BASIC V 6	BASIC version 6 (part 1)
10	B Tackaberry	Plotting 3D surfaces	Surfaces - plotting them in SBAS
14	F Parkes	Recovering files	File recovery off bad disks
17	Bill Aitken	A 380Z disassembler	Disassembler for the 380Z
20	Max Westby	SIGNON - identify M/C	Identifying BASIC's with M/C add-ons
22	Clive Williams	Saved from disaster	OLD command for the 380Z
26	R P Lister	BASIZE	STAT from BASIC
31	Nick Cutler	PUNCTURE.BAS	Punctuation program
34	Aunt Agony	Dear Aunt Agony	Agony Aunt column!
37	Nick Wedd	Auto ^F in TXED	CTRL F perils in TXED
38	J Illingworth	Scrolling, Communicate	Terminal emulation subroutine
41	Nick Wedd	Interrupts in the 380Z	Interrupts in the 380Z!
45	T O Pressed	Library of functions	SBAS functions - what machine am I on?
48	Nick Wedd	Introduction to EMTs	EMTs - an Introduction
51	Peter Clarke	Read Using and readers	Review of Mr Glassborows programs
54		Sorting competition	Competition results - sorting
63	Spencer Instone	letter	SMP A-level suite of programs
64	Clive Williams	String Search Utility	String Search and replace for BASIC 5
70	The Vole	480Z ROS secrets scoop	480Z ROS secrets released!
72	Nick Wedd	User Group Library	Library details
73	Peter Jowett	Drawing polygons	Polygon drawing in HRG

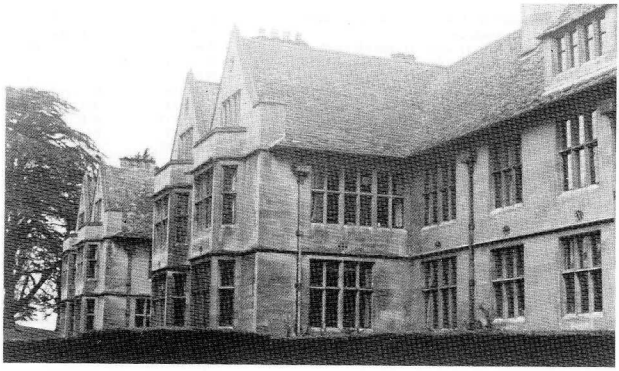
Here is a subject orientated view of that same list (we'll pack the pages somehow!).

Subject	Where it is	Author	Title
380Z/480Z comparison	RU5 p108	Jonathan Darby	Is the 380Z a good buy?
480Z ROS secrets released!	RU7 p70	The Vole	480Z ROS secrets scoop
480Z as a printer buffer	RU6 p45	Robert Wills	Using a link
480Z sound without escape sequences	RU4 p16	S P A	Sound for the 480Z
80 line HRG text	RU5 p105	James Watson	Text Format checker
ALGOL - listing programs from it	RU4 p72	P J Butt	LIST.ALG
Adding CALLs to BASIC (quick version)	RU5 p57	Nick Wedd	Speeding up CALLs
Adding machine code CALLS to BASIC	RU6 p75	W J Shore	Adding things to BASIC
Administration program for DES and ISIS	RU5 p88	D G Whitehouse	School Administration
Agony Aunt column!	RU7 p34	Aunt Agony	Dear Aunt Agony
Ambiguous file name handling from BASIC	RU4 p74	Harry Morten	Super directory
Assembly language - an introduction	RU5 p7	Nick Wedd	Understanding Assembler
BASIC - adding machine code routines	RU3 p52	J M Ridge	Extending BASIC
BASIC - using it as a text editor	RU3 p47	Derek Wickham	BASIC as an editor
BASIC version 6 (part 1)	RU7 p4	G Dale	Extended BASIC V 6
BCPL - reply to review	RU6 p9	John Richards	BCPL reply
Battleships - a BASIC program	RU3 p42	P A Watmough	Batships
Bearings with a BASIC program	RU5 p76	P A Watmough	BEARINGS
British Library project	RU3 p33	ASLIB	ASLIB
CP/M - finding a bit of room	RU6 p78	Nick Wedd	Use of odd corners
CP/M Library - brief review	RU3 p39	Peter Jowett	CP/M User Library
CTRL F perils in TXED	RU7 p37	Nick Wedd	Auto ^F in TXED
Chairman's report	RU2 p3	Bill Tagg	Chairman's report
Characters - defining your own	RU4 p49	J H Ridge	Character Building
Colour improvements by swapping leads	RU6 p36	Nick Wedd	Swapping colours
Competition results - one-liners	RU7 p2		One line programs
Competition results - sorting	RU7 p54		Sorting competition
Controlling a camera	RU4 p33	Nick Cutler	Introducing Captain Z
Copying a program onto several disks	RU6 p87	Bill Aitken	MCOPY.ZSM
Copying and printing methods	RU4 p67	Nick Cutler	Multiple hard copies
Copying disks from BASIC	RU6 p53	Jon Sheppard	Copying discs
Counter-timer chip - using it	RU5 p36	James Watson	Using the CTC
DATE.SYS file - updating it	RU5 p24		DATE.B
DIR - adding free disk space info.	RU5 p106	Bill Aitken	Patching CP/M
DIR - putting output into columns	RU6 p92	H H Jones	Enhancing DIR display
Dating a disk	RU5 p41	P G Lewis	Computer dating service
Debugging tool for BASIC	RU1 p4	J Townsend	Keyswap
Disabled people - using computers	RU2 p10	Helen Townley	Computers and Disabled
Disassembler for the 380Z	RU7 p17	Bill Aitken	A 380Z disassembler
Disk-copying: for 5" disks	RU3 p28	J M Ridge	Fast disk copying
Disk-copying: for 8" disks	RU3 p17	Max Westby	Fast full disk copy
Drawing CALL from BASIC	RU6 p13	D R Ellis	Why not CALL "DRAW"?
Dumping HRG and LRG on a Diabolo	RU5 p26	Max Westby	Diabolical Hard Copy
Dumping HRG on an Epson	RU6 p64	A D Petford	Hardcopy graphics
Dumping LRG on the Epson	RU6 p82	A P Bray	Epson LRG dump
Dumping the screen to a printer	RU4 p43	H H Jones	Extending BASIC
Dumping the screen to an Epson MX80	RU5 p58	Don Smith	Copy screen to printer
EMTs - an Introduction	RU7 p48	Nick Wedd	Introduction to EMTs
Equation solution with Newton Raphson	RU4 p38	David Williams	Polynomial
Executive committee alternative minutes	RU4 p8	Andy Webb	A O B
FORTRAN - adding CALL EMT()	RU3 p36	A Turner-Smith	EMT calls from FORTRAN
FORTRAN - various useful additions	RU4 p	Wilfrid Sollom	Fortran Supplement
File recovery off bad disks	RU7 p14	F Parkes	Recovering files
Find the lady - BASIC program	RU3 p67	P A Watmough	Findlady
Front Panel, a simpler one	RU1 p2	Angela Hirst	Mini-Z
GOing back into your program	RU4 p28	S P A	Zero length files
Graphics - facilitating their use	RU4 p64	P G Lewis	PIC
Guidelines for Programmers	RU3 p2	Nick Cutler	Guidelines
HRG - speeding it up using cache	RU5 p50	J M Ridge	Images of Memory
Handicapped students at an F E College	RU3 p27	Judith Gordon	Helping the handicapped
Hertfordshire's standards	RU2 p11		Micro Standards
IBM 3740 to CP/M conversion	RU1 p8	Nick Cutler	IBM 3740
Identifying BASIC's with M/C add-ons	RU7 p20	Max Westby	SIGNON - identify M/C
Indenting BASIC listings	RU5 p94	J H Ridge	Pretty printing
Indexes from BASIC	RU6 p55	Philip Hepburn	Letter from Gwent
Inputting from CP/M without EMT's	RU5 p12		Input routine
Inputting to a file using PIP	RU5 p17	Jim Reddaway	Typing to disk
Interrupts in the 380Z!	RU7 p41	Nick Wedd	Interrupts in the 380Z
Joystick interface	RU2 p15		Joysticks
LOGO - the Croydon Turtle Graphics	RU4 p10	Malcolm Neave	Croydon LOGO
Labelling disks by filenames	RU2 p21	George Dawson	Labelling disks

Read Using Index

Languages on the 380Z - a comparison	RU6 p18	D Peacock	A language comparison
Library details	RU7 p72	Nick Wedd	User Group Library
Library facilities for the User Group	RU6 p11	Richard Syrett	RML User Group Library
Listing long files in 2 columns	RU5 p71	Bill Aitken	Epson Utilities
MID$= - putting it into BASIC 5	RU2 p9		MID$()=
Micros in Computer Education	RU1 p10	Morfydd Edwards	MICE
Micros in Computer Education news	RU2 p34	Morfydd Edwards	MICE
Micros in Schools awareness pack	RU5 p112	W E C	Not a review
Microsoft's compiling BASIC	RU5 p42	Nick Cutler	Compiling MBASIC
Music board for a 380Z	RU5 p82	Steve Thomas	From Bach to Byte
NIM - a program that 'learns'	RU2 p25		Sticks
Nascoms - using them as terminals	RU2 p31	Pat Crabb	The Nascom and 380Z
OLD command for the 380Z	RU7 p22	Clive Williams	Saved from disaster
Oscilloscope receiving 380Z output	RU6 p74	Nick Cutler	Output to a scope
PIO - curing a resetting bug	RU3 p24	DGS/CEM	Resetting the PIO
PIO - interconnections	RU2 p20	David Coates	PIO Plagiarised
Polygon drawing in HRG	RU7 p73	Peter Jowett	Drawing polygons
Polyphonic music form the 380Z	RU2 p19	David Mitchell	Four voice polyphony
Polytechnic of the South Bank courses	RU4 p48	B E Weller	Scientific Computing
Printer initialisation from CP/M	RU6 p69	Nick Wedd	Initialising a printer
Printing Microquery output nicely	RU6 p80	J G Hunniford	PRINTSET
Privacy codes - how to overcome them	RU2 p7	N W Richards	XDB Privacy codes
Punctuation program	RU7 p31	Nick Cutler	PUNCTURE.BAS
Research User Group report	RU6 p39	Chris Brown	Research User Group
Restoring badly erased files!	RU4 p54	P Jowett	Disk Crash Recovery
Restoring erased files	RU1 p12		Restoring files
Restoring erased files on 8" disks	RU2 p17	J H Carpenter	Restoring files
Review defence of Tonia Cope	RU6 p79	Briget Harrison	Computing using BASIC
Review of 380Z terminal software	RU5 p81	Nick Wedd	Linkup
Review of Laboratory Microcomputer	RU5 p118	Nick Cutler	Laboratory Micro
Review of Mr Glassborows programs	RU7 p51	Peter Clarke	Read Using and readers
Review of RML LOGO	RU6 p37	Peter Butt	LOGO - first lessons
Review of Richards Computers BCPL	RU5 p18	John Ward	BCPL
Review of SBAS - Structured BASIC	RU3 p10	P Smith	SBAS
Review of Tonia Cope's book	RU4 p15	P Smith	Computing using BASIC
Review of Tonia Cope's book	RU5 p114	Solomon Tomides	An alternative review
Review of the Network	RU6 p41	Richard Naylor	First chain experiences
Review of the Network!	RU6 p43	Jonathan Darby	Review of the Network
SBAS - example programs	RU4 p41	S P A	Structured BASIC
SBAS functions - what machine am I on?	RU7 p45	T O Pressed	Library of functions
SMP A-level suite of programs	RU7 p63	Spencer Instone	letter
STAT - how it works	RU4 p46	Michael Wilson	CP/M
STAT both sides and print them	RU5 p52	W M Aitken	PRINTDIR.ZSM
STAT both sides of a disk	RU5 p70	Bill Aitken	Epson Utilities
STAT from BASIC	RU7 p26	R P Lister	BASIZE
STAT from BASIC - 8" systems	RU4 p70	Harry Morten	Incorporating STAT
STAT from BASIC version 5	RU3 p63	J M Ridge	STAT from BASIC
STAT from XDB	RU2 p36		Disk STAT from BASIC
Secretary's report!	RU2 p5	Derek Esterson	Secretary's report
Single siding your disks	RU5 p116	Mike Blandford	Double your disks
Social science program (light-hearted!)	RU2 p29		Deviance
Sorting in BASIC with a shell-sort	RU4 p62	Nick Cutler	Shell Sort
Sorting in SBAS with a quicksort	RU4 p61	S P A	SBAS2 Quicksort
Sorting multiple files with CBASIC	RU5 p64	Robert Wills	Sorting large Databases
Sound article RU5 - correction	RU6 p97	Nick Wedd	Correction
Sound from a 380Z	RU6 p70	J C Ward	Make your 380Z noisy
Sound from an 80 ch 380Z (correction)	RU6 p97		
Sound from an 80 character 380Z	RU5 p38	Nick Wedd	380Z Sound production
Space Invaders!	RU2 p12	James Watson	Utility Software
Stack - moving it out of the way	RU6 p81	Nick Wedd	Moving the stack
String Search and replace for BASIC 5	RU7 p64	Clive Williams	String Search Utility
Subroutines - naming them in BASIC	RU3 p16	S P A	BASMOD
Surfaces - plotting them in SBAS	RU7 p10	B Tackaberry	Plotting 3D surfaces
TXED macros prepared on the screen	RU6 p17	Nick Wedd	TXED: preparing a macro
TXED quirks	RU6 p54	Anon	OBOC
Terminal emulation subroutine	RU7 p38	J Illingworth	Scrolling, Communicate
Text typed straight into a file from CP/M	RU4 p18	J M Ridge	Simple Input
Turnkeying disks	RU1 p11	Brian Drury	Make CP/M work for you
Turnkeying disks	RU4 p22	Nick Cutler	Turnkey Systems
Turnkeying disks	RU4 p29	Nick Wedd	Turnkey systems
Turnkeying disks	RU5 p110	D J Smart	AUTOBOOT.ZSM
Turnkeying disks again (FDS)	RU5 p100	H Morten	Yet more on Turnkey
Wordstar - Practical advice	RU6 p61	Philip Hepburn	Wordstar
XDB - handling all errors properly	RU2 p23		Crash proofing disks
XDB - restoring the private parts!	RU3 p51	Peter Jowett	Secure Privacy
XDB - skip Highest memory, disable CTRL-T	RU6 p7	N R Wardell	Patches for XDB
XDB directory file	RU5 p69	P G Lewis	Superdirectory

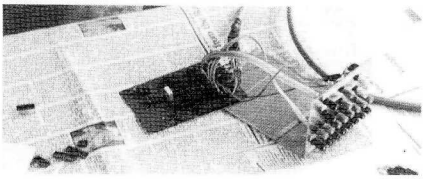
Snaps from the User Group Workshop Weekend 24th-25th March



Deep in heart of Blagdon...

Frenzied knitting is taking place...

Split second timing is needed when it comes to the boil



The finished time machine.

Please turn over for an important announcement...

RESEARCH MACHINES

April 1984

Sales Information

New 16 bit computer

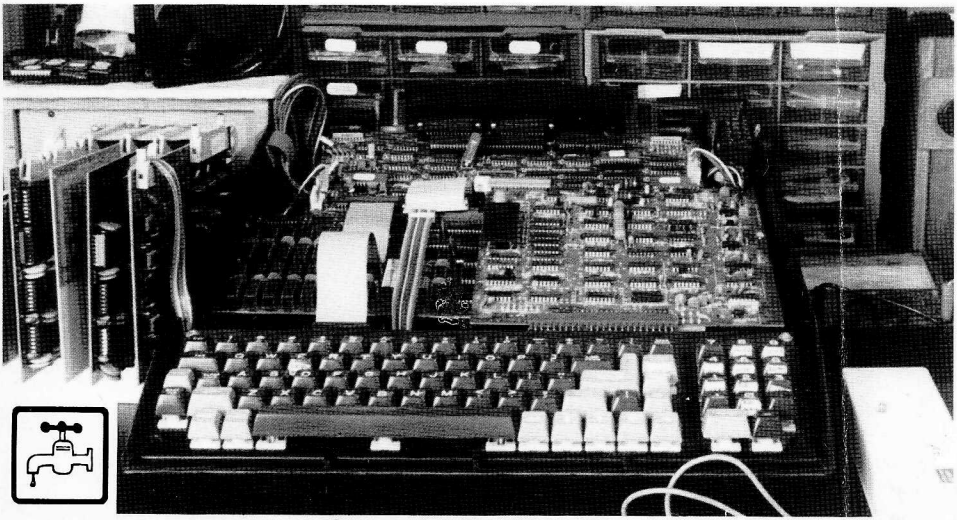
At long last we have unveiled our latest and greatest computer. At a meeting of the user group executive committee at the beginning of April we announced details of the new Research Machines 'Mycroft 580Z' 16 bit computer range and demonstrated a prototype machine for a short while. the reception was fantastic! The years of careful design have resulted in a product that we are very proud of and received the thorough approval of the committee.

We believe that in the Mycroft we have created the first truly general purpose computer. We are not aiming it at a particular market because it will perform brilliantly in all markets. The research user will find the expandable power of the Mycroft unbeatable. In education the ease of use and friendliness will be the Mycrofts' strong points. The business user is well supported by applications packages written using our new business applications language. all this for the TRP (typical retail price) of £499.59 makes the Mycroft a personal computer with a place in the home.

Expandable Power

At the heart of the Mycroft 580Z are two powerful iAPX181 processors. This relatively recent chip heralds the beginning of a new era in computing. The iAPX181 has a 16 bit internal data bus; however, the revolutionary power of this chip lies in the 1 bit high speed serial external data bus. This is almost literally one wire which connects the two CPUs and links the peripheral and interface controllers. This system gives the Mycroft exceptionally cheap expansion capabilities. Adding one extra processor doubles the Mycroft's power - since we supply Mycroft with two processors it may yet become known as the 'two bit computer'!

Because of the nature of the data bus it has been called the 'hardware pipeline'. Extra processors and other compatible circuitry simply tap onto the pipeline. The pipeline is almost indistinguishable from a serial network system and has already been used as such. The final production machine will have inbuilt compatibility with the Chain network. The Mycroft is unique in using the Rugby radio 'Time Coded Transmissions' to clock its internal circuitry. Not only does this mean that it tells the right time when you switch it on but it also ensures the synchronization of all 580Zs in the country! This makes the cost of a reliable long distance high speed data links very low indeed since no extra wires for clock signals are needed.



The cover is reprinted from part of a sales sheet that is about to be released. Since the remainder of it was even more technical, I have cobbled together these notes from various people's impressions of the machine and the bumph that we were showered with at the meeting. The photo shows one of the prototype Mycrofts being deep probed.

'Eyecon' operating system

The whole of the Mycroft system depends upon Eyecon. This is a Multi-tasking/Concurrent operating system (that is, it can run lots of programs at once). It is based on a system of data streams called pipes. Eyecon is effectively the software equivalent of the Mycroft's hardware pipes and to the programmer there need be no distinction. Programs simply tap onto pipes as necessary to communicate with each other or to use peripherals. Data put into the pipeline can be picked up later by another program. Additional peripherals appear as more taps on pipelines. Storage is implemented by enabling data to be pooled in tanks along the pipeline.

The 'user metaphor' is completed by providing the PRUNE utility which flushes the system removing unwanted data that might eventually block the pipelines. Special pipes with the 'Semantic Bends' (S bends) are used to communicate between concurrent instances of the same process. The PLUMB utility is provided for systern maintenance: unfreezing pipes, getting rid of gurgling noises, and so on. The symmetry of the Mycroft ensures that the Eyecon pipelines are entirely compatible with the pipelines used by the computer controlled water cooling and so any two 580Z computers can communicate via the local water board.

Hardware

The production model of the Mycroft is, of course, going to be ruggedly constructed and is designed to fit into a 21 inch rack mounting case. The front panel sports two keyholes. One of the keyholes is for a digital key that can be read by the computer. Software companies will make use of the unique key codes to prevent software from running on other machines. This facilitates true 'dual key' operation.

The front panel that we are used to in RML computers, that is, the software front panel has actually been implemented with hardware in the Mycroft. Encouraged by the enthusiastic response to the flashing red light on some of the 380Z boards RML have gone the whole hog. The front panel is in fact on the side of the machine, a cover slides aside to reveal lots of flashing lights and a row of 16 switches. This 'user hostile systems interface' is deliberately designed to discourage the casual user from tinkering around with the computers internals. In contrast there is a software back panel which keeps an eye on all of the Mycrofts dealings with the outside world telling you when your pipes are blocked or taps are leaking. The hardware back panel (which is actually underneath the machine) contains the very simple provisions for interfacing Mycroft. One 4mm banana socket for the network and any other devices that tap onto the pipelines. Additionally there are several 3 pin Mains sockets for control purposes - RML envisage these being used for real time control of fan heaters, coffee percolators, and the like. Via the operating system there are many ports available and a wide selection of wines and spirits is also planned. The RML personal touch shows: inside the case are moulded the names of the 422 Duran-Duran groupies on the production team.

Software

Initially the Mycroft will be supplied with BASICSG2 versions 5 and 6. A version 7 of this BASIC is planned which includes a REPEAT-UNTIL loop. Better than BASIC however is the SEEDS business application language. Written in SEEDS is the Peartree spreadsheet program. Famous for its telling motto "When it hits the fan it's time to climb the Peartree", this spreadsheet will help you plan the best methods of spreading rumour, dissent, scandalous accusations, lies and propaganda. RML have been using it for some time now. A free partridge goes to every 1000th purchaser of the spreadsheet.

The database management system to be supplied has a very friendly and approachable user interface metaphor that anyone with some experience of office practice will be abe to relate to. The system structure groups all data into metasubjects called **drawers.** Each drawer is subdivided into a number of **files** into which the raw data is placed. For instance, a memo might be filed by first pulling out the correct drawer, selecting an appropriate file and stuffing the memo into it. A file that gets too full can be emptied by moving it over to the wastepaper basket and turning it upside down. Throughout this process, no actual typing has to take place. All functions are performed by moving to the appropriate icon in the metaphor and, selecting, pulling or turning it. Rather than using expensive bit-mapped graphics to achieve this which some of the slower secretaries may not fully grasp, RML provide the system icons on a 1.5m box with four drawers and an optional number of files.

Peripherals

A large range of Mycroft peripherals are planned. Shortly after the initial release a monitor and keyboard (with soft keys "that ooze friendliness") should be available. A fixed disk system option (with no moving parts to go wrong) just taps onto the pipeline.

The Monitor is, in fact, a projecting television of some quality. This remarkably sensible approach means that the monitor can be used for almost any application. The size of the image produced depends on the monitor's distance from whatever is being used as the screen. This makes it invaluable in the classroom or lecture theatre where you just use a handy wall painted with silverdiapropanol sensitizer. On the train you would simply paint the person sitting opposite to you. Enormous possibilities are opened up by introducing shadows into the projected picture, The user group committee had no end of fun doing this but had to stop when the machine crashed just as a small dog and a ferocious butterfly were about the eat the marketing manager. Well, there's devotion for you!

The keyboard has audio feedback to help touch typists. As was demonstrated the sounds produced are fully programmable. The programmer who set up the prototype clearly has a very affectionate nature judging by the discraceful sounds we heard. The keyboard is fully redefinable and to overcome this problem RML supply a small (black or cream) paint pot to blank out misleading symbols on the keys such as the letters.

Documentation

"Is planned".

All statements on the previous page should be ignored.